micro informatica srl



INFORMATICA PENTRU ELEVI

*micro*Informatica

Editat sub egida Comisiei Naţionale de Informatică

Florian Boian
Ioana Chioreanu
Gheorghe Coman
Militon Frenţiu
Sever Groze
Kása Zoltán
Teodor Toadere
Leon Ţîmbulea

Informatică pentru elevi

Cluj-Napoca
1992

Coperta

designer Liviu Derveşteanu

Editor

prof. univ. dr. Emil Muntean

1. SISTEM DE CALCUL.

1.1. Elemente de prelucrare automata a datelor.

O caracteristică esențială a secolului al XX-lea o constituie rapida dezvoltare a științei și tehnicii, într-o strînsă cooperare. În aceste condiții, proiectarea diferitelor tehnologii și stabilirea unor soluții optime ale acestora au dus cu necesitate la efectuarea de calcule tot mai numeroase și complexe, imposibil de realizat cu metodele și mijloacele existente anterior. Drept consecință, apariția mijloacelor automate de calcul (calculatorul electronic), în cel de al cincilea deceniu al secolului nostru, a devenit o necesitate, constituind una dintre cele mai mari cuceriri ale gîndirii și tehnologiei umane.

Prin prelucrare de informații se înțelege efectuarea asupra lor a unor transformări care în general pot fi de:

a) introducere (citire) în dispozitivul de calcul;

b) transferări de date dintr-un loc în altul, în vederea organizării lor;

c) operații de calcul (adunări, scăderi, înmulțiri, etc.);

d) extragerea (tipărirea) rezultatelor.

După gradul de participare a omului în procesul prelucrării, după tipul muncii efectuate și după echipamentul utilizat, metodele de prelucrare pot fi: manuale, electromecanice și automate. Prelucrarea automată este net superioară celorlalte datorită preciziei și vitezei de lucru.

Ceea ce deosebește radical prelucrarea automată a informațiilor de celelalte tipuri de prelucrări este faptul că în cazul acesteia intervenția omului, exceptînd cazurile de forță majoră, este necesară o singură dată, la început, atunci cînd dispozitivului de prelucrare i se dă împreună cu datele de prelucrat și programul de lucru.

Dispozitivul care efectuează o astfel de prelucrare are următoarele părți componente:

- unitatea centrală;
- memoria internă;
- dispozitive periferice.

Unitatea centrală are drept scop decodificarea și execuția instrucțiunilor furnizate de utilizator. Ea este compusă din:

- unitatea de calcul, care efectuează operațiile propriu-zise;

- unitatea de comandă, care dirijează întreaga activitate a dispozitivului.

Memoria este alcătuită din elemente bistabile, dispozitive fizice capabile să se prezinte în două stări distincte. Asociind acestor stări cifra 0, respectiv 1, fiecare dispozitiv poate reprezenta o cifră binară sau **bit** (binary digit), considerat a fi unitatea elementară de informaţie. Orice memorie are o structură de organizare. Cel mai frecvent este împărţirea ei în locaţii (celule), o locaţie fiind alcătuită dintr-un număr oarecare de elemente bistabile. Fiecărei locaţii i se asociază o adresă cu ajutorul căreia locaţia respectivă este identificată.

Numărul elementelor bistabile ce alcătuiesc o locaţie determină lungimea ei şi diferă de la calculator la calculator. Lungimea minimă a unei locaţii adresabile este, de obicei, de 8 biţi, iar locaţia se numeşte *octet*.

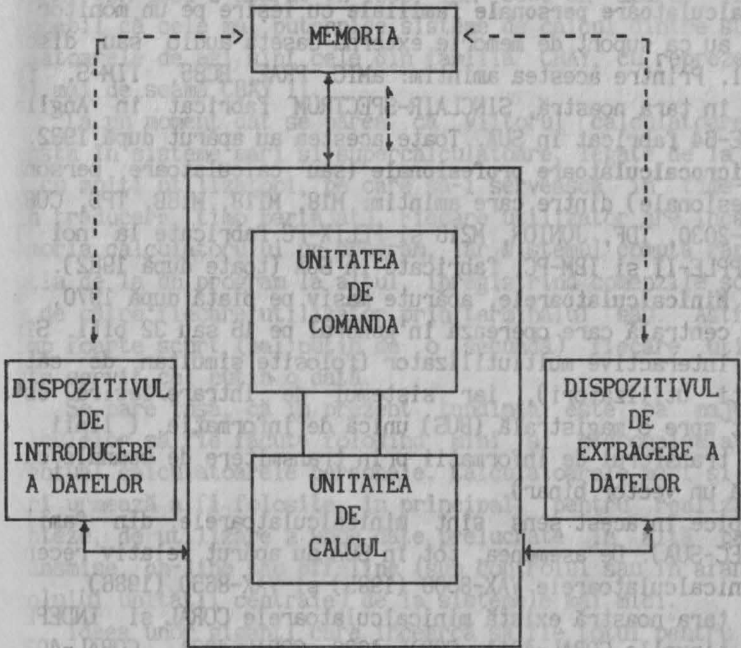
Dimensiunea memoriei unui calculator se exprimă în *K octeti*, sau în *M octeti*, unde 1 K (Kilo octet) = 1024 octeţi, iar 1M (Mega octet)= 1024 K.

Dispozitivele periferice au rolul de a citi informaţia iniţială de pe anumite medii externe, de a stoca informaţii şi de a extrage rezultatele sub o formă dorită de către utilizator.

Legătura dintre aceste dispozitive este prezentată în figura de mai jos, unde prin linii continue s-a marcat impulsul de comandă, iar prin linii discontinue fluxul de informaţie (transferul de date).

Din această figură rezultă că pentru o prelucrare automată folosim mai mult decît un calculator; folosim un sistem de prelucrare automată a datelor (SPAD), calculatorul fiind doar o componentă a acestuia. Cu toate acestea, este răspîndit obiceiul de a denumi sistemul de prelucrare, calculator electronic, lucru ce va fi făcut şi de către noi în continuare.

Cele trei unităţi funcţionale se pot afla într-un singur bloc fizic sau ca unităţi fizice distincte interconectate prin cabluri de legătură. Ele constituie partea fizică a calculatorului denumită şi HARD (denumirea provine de la cuvîntul englezesc hardware).



Toate aceste dispozitive rămîn inerte în lipsa unor programe puse la dispoziția utilizatorului de către firme specializate și care alcătuiesc **sistemul de operare** sau SOFT-ul calculatorului (denumirea provine de la cuvîntul englezesc software).

Partea fizică împreună cu sistemul de operare alcătuiesc **sistemul de calcul**.

1.2. Tipuri de sisteme de calcul.

Familiile de calculatoare existente astăzi în lume, ca și la noi în țară, pot fi împărțite în mai multe clase:

1) **Microcalculatoare**. Acestea folosesc ca unitate centrală un microprocesor (dispozitiv realizat cu un număr oarecare de componente integrate capabil să execute instrucțiuni, format dintr-o unitate de comandă și o unitate de calcul) care operează pe 8, 16 și în ultima vreme pe 32 de biți.

Microcalculatoarele sînt sisteme interactive monoutilizator (prelucrarea este dirijată în mod nemijlocit de la un terminal de către o singură persoană). Ele se împart, la rîndul lor, în:

-**Calculatoare personale familiale** cu ieșire pe un monitor TV și care au ca suport de memorie externă caseta audio sau discul flexibil. Printre acestea amintim: aMIC, PRAE, HC85, TIM-S, fabricate în țara noastră, SINCLAIR-SPECTRUM fabricat în Anglia, COMODORE-64 fabricat în SUA. Toate acestea au apărut după 1982.

-**Microcalculatoare profesionale** (sau calculatoare personale profesionale) dintre care amintim: M18, M118, M18B, TPD, CUBZ, DIAGRAM-2030, TDF, JUNIOR, M216 și FELIX-PC fabricate la noi în țară, APPLE-II și IBM-PC, fabricate în SUA (toate după 1982).

2) **Minicalculatoarele**, apărute masiv pe piață după 1970, au unitate centrală care operează în general pe 16 sau 32 biți. Sînt sisteme interactive multiutilizator (folosite simultan de către mai mulți utilizatori), iar sistemul de intrare/ieșire este orientat spre o magistrală (BUS) unică de informație (linii ce asigură transferul de informații prin transmitere de semnale care codifică un vector binar).

Tipice în acest sens sînt minicalculatoarele din familia PDP11(DEC-SUA). De asemenea, tot în SUA au apărut relativ recent, superminicalculatoarele VAX-8600 (1985) și VAX-8650 (1986).

În țara noastră există minicalculatoarele CORAL și INDEPENDENT de tipurile CORAL-4011, CORAL-4030, CORAL-4031, CORAL-4021, respectiv I-100, I-102F, I-106. În prezent există preocupări pentru realizarea și la noi în țară, a unor minicalculatoare de tip VAX care să opereze pe 32 de biți.

Minicalculatoarele au un preț de cost rezonabil. Astfel, spre exemplu, un minicalculator CORAL-4030 cu dotare completă costă în jur de 3 milioane de lei (în 1988).

3) **Calculatoarele medii-mari** au apărut după 1950. Operează pe cuvinte de 32 biți, au memorie mare, dar sînt deosebit de scumpe. Familia dominantă a acestei categorii de calculatoare este IBM-360. În țara noastră astfel de calculatoare sînt FELIX C-256, FELIX C-512, FELIX C-1024 și FELIX C-5000. Orientativ, un calculator FELIX C-256 cu o dotare completă a costat, la nivelul anului 1979, aproximativ 30 milioane lei. În prezent se fabrică din această categorie numai FELIX C-5000, după o tehnologie nouă. El are performanțe mai mari decît predecesorii lui și un preț de cost mai scăzut: 10 milioane lei.

4) **Supercalculatoarele** au o memorie internă de mare capacitate, eventual mecanisme de memorie virtuală, au operanzi pe cuvinte de 32-64 biți și sînt sisteme multiprocesor. Din această familie amintim IBM-370 cu seriile 145, 158, 168, CDC 6600 și CDC

CYBER, IBM 4381-3 apărut în 1985, IBM 3083-JX apărut în 1983. Probabil că cele mai puternice sisteme de calcul dintre supercalculatoarele de azi sînt cele din familia CRAY, cu reprezentantul cel mai de seamă CRAY II.

La un moment dat se părea că viitorul calculatoarelor ar consta în sisteme mari și supercalculatoare, legate de la distanță cu mulți utilizatori, pe care să-i servească în *time-sharing* (în traducere, timp partajat). Fiecare utilizator are încărcat în memoria calculatorului un program, iar sistemul comută rapid execuția de la un program la altul, înregistrînd comenzile solicitate de către fiecare utilizator prin terminalul său. Astfel, în timp foarte scurt (mai puțin de o secundă) fiecare utilizator este servit cel puțin o dată.

Se pare însă, că în prezent tendința este ca majoritatea calculelor să fie făcute folosind mini și microcalculatoarele, eventual calculatoarele personale. Calculatoarele mari și foarte mari urmează a fi folosite, în principal, pentru realizarea de sinteze, de utilizare a unor date prelucrate în altă parte și transmise, **on-line** sau **off-line** (sub controlul sau în afara controlului unității centrale) de la sistemele mai mici.

Ideea unor giganți care încearcă să fie totul pentru toți se pare că nu mai are sorți de izbîndă.

Se poate vedea în acest sens comparația între rețelele superminicalculatoarelor din familia VAX (rețeaua VAX Cluster, care tinde să integreze și calculatoare personale în rețea) și propunerea IBM de utilizare a unui supercalculator lucrînd sub sistemul de operare MVS, în locul unei rețele ([8],pp.51-56).

1.3. Structura si functionarea unui microcalculator personal.

Prezentăm în continuare structura unui microcalculator personal. El se compune din: memorie, unitate centrală și periferice (dispozitive de intrare/ieșire).

Memoria are sarcina de a stoca informații (date și program) într-o formă bine stabilită. Există diverse tipuri de memorii, prezentarea lor fiind aproape imposibilă și lipsită de interes. Ele se clasifică în:

1. **Memorie RAM** (Random -Access -Memory) folosită ca memorie de lucru, în care se poate atît scrie cît și citi. Nu poate fi folosită pentru a păstra informații pe termen mai lung fiind volatilă (conținutul ei se pierde în momentul întreruperii sursei

electrice).

2. Memorii ROM (Read - Only - Memory) care sînt nevolatile însă care nu pot fi decît citite. În general, o memorie ROM este programată direct de firma producătoare.

Majoritatea calculatoarelor au memoriile interne intre 64 KO și 16 MO.

Unitatea centrală are sarcina de efectuare a calculelor, gestiunea memoriei și realizarea unor funcții logice. Conține unul sau mai multe registre acumulator folosite pentru păstrarea datelor aduse din memorie, efectuarea operațiilor asupra acestor date și retransmiterea lor memoriei.

În afara acestor registre acumulator unitatea centrală dispune și de alte registre ce îndeplinesc funcții speciale. Astfel, există un registru ce poate păstra o adresă curentă în memorie. Dimensiunea acestuia determină capacitatea memoriei care poate fi direct adresată de unitatea centrală.

De asemenea ea mai conține un registru în care se memorează instrucțiunea ce trebuie executată, conținutul lui fiind interpretat de către unitatea centrală și un registru, numit contor de program, care adresează instrucțiunea ce trebuie să fie executată în continuare.

Unitatea centrală funcționează, în general, pe baza principiilor magistralelor comune prin care datele, reprezentate în general pe 8 sau 16 biți, sînt transferate la și de la unitatea centrală.

Perifericele sînt echipamentele ce permit transmiterea informațiilor și stocarea rezultatelor.

Cel mai des periferic utilizat este tastatura sau claviatura, care seamănă foarte mult cu cea a unei mașini de scris.

Legatura între tastatură și microprocesor, în vederea transmiterii datelor se face astfel: fiecărei litere, cifre sau caracter special i se asociază o valoare zecimală care apoi este convertită în binar pentru a fi "înțeleasă" de microprocesor. Codul cel mai frecvent utilizat în acest scop este codul ASCII (American Standard Code for Information Interchange), formulat în 1963.

Pentru efectuarea dialogului între utilizator și calculator precum și a vizualizării rezultatelor se folosește un dispozitiv de afișare. Acesta poate fi un dispozitiv de DISPLAY sau un televizor care, în general, permite afișarea a 80 de caractere pe o linie a ecranului.

Deoarece memoria calculatorului este limitată și volatilă, pentru înlăturarea acestor neajunsuri se folosesc memorii externe (auxiliare).

Cel mai uzual suport de memorie externă este banda magnetică (caseta), iar ca periferic poate fi folosit casetofonul. Dezavantajul acestora constă în viteza redusă de lucru, datorată accesului la informațiile depozitate.

Unele calculatoare personale folosesc ca și memorii externe discul flexibil. Utilizînd ca periferic un cititor de disc flexibil, viteza de incărcare a programelor crește foarte mult. Pe un astfel de disc flexibil se poate stoca o informație cuprinsă între 128 și 1024 KO.

Pentru eventuala tipărire a rezulatelor se utilizează imprimanta.

1.4. Limbaje de programare.

Un sistem de prelucrare automată este capabil să rezolve o anumită problemă numai dacă, pe lîngă datele problemei i se comunică și programul de lucru (algoritmul de rezolvare a problemei respective). Comunicarea acestor informații trebuie făcută într-o formă perceptibilă lui, într-un limbaj accesibil calculatorului, numit *limbaj calculator sau masina*. Operația de elaborare și descriere a programului pentru un calculator se numește de *programare* și este specifică numai prelucrării automate. Programul scris în limbaj mașină va fi numit *program obiect*.

Deoarece limbajul calculator utilizează numai caractere numerice (cifrele 0 și 1), o instrucțiune a programului obiect apare ca o succesiune de astfel de cifre. Aceasta face ca programarea în limbaj mașină să prezinte o serie de dezavantaje datorită următoarelor motive:

-limbajul este greu de învățat și de reținut;

-corectarea eventualelor erori intervenite în programare se face foarte greu;

-limbajul diferă de la calculator la calculator, ceea ce presupune învățarea unui limbaj mașină pentru fiecare calculator.

Toate acestea au făcut ca operația de programare a unui calculator în limbaj mașină să nu mai fie de mult practicată, ea fiind caracteristică numai primelor calculatoare.

Calculatoarele moderne utilizează în programare limbaje intermediare între limbajul vorbit și cel calculator numite *limbaje*

de programare. Ele sînt limbaje artificiale, construite cu o anumită structură gramaticală și care folosesc anumite caractere pentru exprimarea dorințelor utilizatorului.

Dar orice calculator poate rezolva o problemă numai dacă ea este formulată în limbajul lui. Apare deci necesitatea traducerii unui program scris într-un limbaj de programare, program pe care o să-l numim sursă, într-un program obiect. Datorită structurii simple și precise a limbajelor de programare, operația de traducere a unui program dintr-un limbaj de programare într-un limbaj calculator a putut fi algoritmizată și lăsată în seama calculatorului, care o execută prin intermediul unui program special, program numit *translator* sau *compilator* și care aparține sistemului de operare.

Aproape toate sistemele de calcul posedă programe translatoare pentru mai multe limbaje de programare, iar programarea într-un astfel de limbaj nu necesită cunoașterea structurii interne a calculatorului.

Limbajele de programare se împart în 2 clase:

1) Limbaje de nivel inferior sau de asamblare;

2) Limbaje de nivel superior sau evoluate.

Limbajele din prima categorie depind foarte mult de calculatorul pe care sînt implementate, deci sînt apropiate de limbajul mașină.

Spre deosebire de primele, limbajele de nivel superior s-au dezvoltat în ideea de a fi orientate spre un anumit domeniu de aplicații și de a depinde cît mai puțin de calculatorul pe care sînt implementate.

În afara criteriului de clasificare de mai sus mai există și altele. Printre acestea unul este legat de modul de lucru la calculator. Astfel, indiferent că este de nivel inferior sau superior, limbajele de programare pot fi:

a) Limbaje conversaționale;

b) Limbaje neconversaționale.

Limbajele conversaționale se caracterizează prin realizarea unei conversații între utilizator și calculator atît în timpul traducerii programului cît și în timpul execuției lui. În acest scop sînt necesare dispozitive terminale cuplate la calculator, de tip display, care permit comunicarea programului direct calculatorului, fără a mai fi necesară înregistrarea lui pe alte suporturi externe.

Avantajul unor astfel de limbaje constă în faptul că, prin

sistemul de conversaţie ce se stabileşte,se creează posibilitatea corectării imediate a eventualelor erori apărute în program,erori semnalate de către sistem.

Limbajele neconversaţionale nu permit intervenţia utilizatorului în timpul compilării şi nici în timpul execuţiei programului. Acesta este modul de lucru tradiţional, existent la majoritatea sistemelor mai vechi de calcul printre care şi a calculatorului romanesc FELIX C-256.

În cazul acestora întregul program scris în limbajul de programare se înregistrează în prealabil pe un suport fizic (cartelă de hîrtie, bandă magnetică, disc magnetic), se comunică sistemului şi din acest moment utilizatorul nu mai poate interveni. După traducerea sau traducerea şi execuţia programului (în funcţie de sistem), utilizatorului i se comunică o situaţie a stării programului: erori de programare, rezultate ale unor calcule etc.

Subliniem trăsătura esenţială a oricărui limbaj de programare: formalizarea strictă şi lipsa oricărei ambiguităţi. Aceasta este determinată de faptul că el este făcut pentru a fi implementat pe un calculator, ori pentru ca acesta să înţeleagă cele scrise de către programator trebuie să fie respectate anumite reguli sintactice.

1.5. Microcalculatorul personal HC-85 si folosirea lui.

Unitatea centrală a microcalculatorului HC-85 este construită cu microprocesorul Z80A şi dispune de o memorie de 64K din care 16 K sînt de tip ROM unde se află interpretorul BASIC (programul care execută rînd cu rînd instrucţiunile Basic), iar 48 K sînt de tip RAM.

Ca şi periferice are un dispozitiv de afişare (un televizor alb-negru sau color de tip PAL acordat pe canalul 10), iar ca memorie externă caseta magnetică acţionată de un casetofon audio obişnuit.

Părţile sale componente sînt legate între ele prin diferiţi conectori:

a) de alimentare a calculatorului cu o sursă de 9V;

b) de interconectare cu un televizor;

c) de conectare a unui casetofon în cazul în care se doreşte încărcarea unui program de pe casetă sau salvarea lui pe casetă;

d) de extensie, folosit la legarea unei imprimante în vederea listării programului.

Pentru introducerea informaţiilor în unitatea centrală şi afişarea lor pe ecran se foloseşte o claviatură foarte asemănătoare cu a unei maşini de scris. Tastele ei conţin caractere literale şi numerice precum şi tastele CR (Carriage Return sau Enter), CS (Caps Shift), SS (Symbol Shift) şi spaţiul (tasta de culoare albă). Acţionarea unei taste duce la afişarea pe ecran a unui simbol simplu (caracter literal, numeric sau special), sau a unui cuvînt cheie (numele unei funcţii).

Pentru obţinerea tuturor comenzilor şi funcţiilor posibile, unele taste au pînă la 6 semnificaţii diferite, precizarea lor făcîndu-se prin acţionarea tastei corespunzătoare simultan cu una din tastele CS sau SS, în funcţie de modul de lucru.

Microcalculatorul are 5 moduri de lucru, prin care se alege un simbol indicat pe tastă, care are forma

Poz.4		Poz.2
	Poz.1	
Poz.5		Poz.3

Aceste moduri precizate de către un cursor (un dreptunghi luminos) care indică locul de pe linia curentă unde va apare următorul caracter sau funcţia introdusă de la claviatură.

Cursorul descrie caracterele literale K, L, C, E şi G corespunzător celor 5 moduri de lucru. Interpretarea lor este dată în continuare.

a) Modul de lucru K (Keyword = cuvînt cheie) apare cînd se aşteaptă o comandă sau un număr de linie.

În acest caz acţionarea unei taste numerice (0-9) duce la afişarea unui caracter numeric (poziţia 1 din figura), iar acţionarea unei taste literale, la afişarea unei funcţii (cuvint cheie, în poziţia 3 din figură) şi trecerea imediată la modul de lucru L.

Tasta SS acţionată împreună cu o tastă literală conduce la afişarea unui caracter din poziţia 2 a figurii de mai sus.

b) Modul de lucru L (Letters = litere) apare imediat după utilizarea unui mod de lucru K şi este folosit la scrierea caracterelor literale mici.

Prin acţionarea tastei corespunzătoare unui caracter literal se obţine caracterul literal mic, iar prin acţionarea unei taste numerice, un număr (poziţia 1 din figură).

Acţionat împreună cu tasta SS, la caractere literale corespunde poziţia 2, iar la cele numerice poziţia 3 din figură.

Acţionat împreună cu tasta CS, la caracter literal corespunde poziţia 1 (cu majuscule) şi la cele numerice poziţia 4 din figură.

c) Modul de lucru C (Capitals = majuscule) este o variantă a lui L, scrierea făcîndu-se cu caractere literale mari. Se utilizează după modul K prin acţionarea tastei CS şi 2. Poate alterna cu modul L.

d) Modul E (Extended = extins). Se obţine prin acţionarea simultană a lui CS şi SS şi se anulează imediat după folosire.

În acest mod de lucru acţionarea unei taste literale generează caracterul din poziţia 4 şi caracterul de pe poziţia 5 dacă este acţionată împreună cu tasta SS.

Acţionarea unei taste numerice în acest mod de lucru generează o secvenţă de control a culorii sau un caracter de pe poziţia 5 dacă este folosită împreună cu tasta SS.

e) Modul de lucru G (Graphics = grafic) este obţinut prin acţionarea tastelor CS şi 9.

Acţionarea tastelor 1-8 în acest mod de lucru duce la afişarea unui mozaic grafic predefinit, iar a unei taste literale, diferită de V, W, X, Y, Z, la afişarea literei respective. Acţionarea tastelor exceptate conduce la următoarele afişări.

V ⟶ RND
W ⟶ INKEY
X ⟶ PI
Y ⟶ FN
Z ⟶ POINT

Acţionarea tastei zero duce la ştergerea caracterului aflat în stînga cursorului.

Pe ecran afişarea se realizează pe 24 de linii, fiecare linie putînd conţine maximum 32 de caractere. Cele 24 linii sînt împărţite în două părţi:

a) partea întîi, formată din liniile 1-22 inclusiv, folosită la afişarea instrucţiunilor sau a rezultatelor programului. Cînd această parte este plină, apare mesajul "scroll?". Tastarea lui CR (sau a oricărui alt caracter diferit de BREAK) duce la afişarea a încă unui ecran s.a.m.d.

b) partea a doua (liniile 23-24) este folosită pentru comenzi de intrare, linii de program, tipărirea datelor de intrare şi pentru mesaje.

1.6. Comenzile de executie si lucrul cu perifericele.

Pentru efectuarea unor operaţii de manevră cum ar fi lansarea în execuţie a unui program, încarcarea unui program de pe o casetă în memoria calculatorului sau copierea lui din memorie pe o casetă, se utilizează anumite comenzi speciale. Acestea sînt:

a) Comanda:

RUN

folosită la lansarea în execuţie a unui program existent în memoria calculatorului. Comanda poate fi folosită şi sub forma:

RUN număr

unde "număr" reprezintă eticheta unei instrucţiuni din program, caz în care execuţia începe de la instrucţiunea cu eticheta "număr".

b) Comanda:

SAVE

se utilizează în operaţia de obţinere a unei copii a programului aflat în memorie pe o casetă. Pentru aceasta programul trebuie să primească un nume compus din maximum 10 caractere literale şi/sau numerice. Sintaxa comenzii este :

SAVE "nume"

la care calculatorul răspunde prin afişarea mesajului:

Start tape then press any key .

Pregătind casetofonul pentru înregistrare şi acţionînd o tastă a claviaturii (de ex. CR), are loc înregistrarea programului pe bandă. La terminarea operaţiei, pe ecran se afişează mesajul

0 OK

c) Comanda:

VERIFY "nume"

are ca efect verificarea programului înregistrat. Pentru aceasta se poziţionează banda la începutul înregistrării, se dă comanda de verificare şi se tastează CR. În momentul găsirii programului înregistrat, pe ecran apare mesajul :

Program: nume

iar la sfîrşitul programului mesajul:

0 OK

dacă înregistrarea a fost făcută corect. În cazul unei erori de

înregistrare pe ecranul monitorului apare mesajul:

R Tape loading error

caz în care se face o nouă înregistrare.

Menţionăm că dacă poziţionarea casetei a fost făcută mult înaintea programului ce urmează să fie verificat, în timpul operaţiei de căutare a programului specificat, calculatorul afişează numele tuturor programelor pe care le întîlneşte.

d) Comanda

LOAD

este complementara lui SAVE, fiind folosită la transferul în memoria internă a unui program existent pe o casetă. Ea şterge din memoria calculatorului vechiul program şi variabilele sale. Sintaxa comenzii este :

LOAD "nume"

Utilizată sub forma LOAD " " are ca efect încărcarea în memorie a primului program întîlnit pe casetă.

Microcalculatoarele personale HC-85, TIM-S, COBRA, PHOENIX, sînt compatibile cu SINCLAIR ZX-SPECTRUM.

Pentru calculatoarele **HC-85** ce au ca suport extern dischete, comenzile corespunzătoare transferului de programe între memorie şi dischete se schimbă în:

```
LOAD *"d";1;"nume-program"
SAVE *"d";1;"nume-program
```

2. ALGORITMI SI DESCRIEREA LOR.

2.1. Notiunea de algoritm.

Ca și alte noțiuni matematice și noțiunea de algoritm se caracterizează printr-o largă generalitate. Această caracteristică a împiedicat elaborarea unei definiții matematice riguroase, formulată într-un cadru general, existînd chiar opinii de a fi acceptată ca o noțiune primară. În consecință, pentru introducerea conceptului de algoritm se recurge la o prezentare descriptivă, intuitivă, punîndu-se accent pe caracteristicile fundamentale ale acestuia.

Astfel, prin **algoritm** se înțelege o mulțime ordonată și finită de reguli care descriu o succesiune finită de operații necesare rezolvării unei probleme.

Așadar, un algoritm transformă cantități date (datele inițiale sau de intrare) în alte cantități (rezultatele finale sau datele de ieșire) în conformitate cu o mulțime dată de reguli, trecînd printr-un număr finit de etape intermediare (număr finit de pași). Notînd prin *Di* mulțimea datelor de intrare ale unui algoritm **A** și prin *De* mulțimea datelor de ieșire ale aceluiași algoritm, algoritmul **A** definește o funcție care pune în corespondență fiecărui element *x* din *Di* un element *y* din *De*, adică :

$$A : Di \rightarrow De.$$

De exemplu, algoritmul lui Euclid, **AE**, pentru calculul celui mai mare divizor comun a două numere întregi *z1* și *z2*, nu ambele nule, asociază perechii (*z1*,*z2*) numărul întreg *d* egal cu cel mai mare divizor comun al numerelor *z1* și *z2*. Deci **AE** : $ZxZ \rightarrow Z$.

De asemenea, notînd prin **A2** algoritmul de calcul al rădăcinilor unei ecuații de grad efectiv doi:

$$"ax^2 + bx + c = 0"$$

a,*b*,*c* fiind numere reale cu $a \neq 0$, **A2** asociază fiecărui triplet (*a*,*b*,*c*) rădăcinile *x1* și *x2* ale ecuației date, adică perechea (*x1*,*x2*) de numere reale sau complexe, după cum discriminantul ecuației este >= 0, respectiv < 0. Așadar avem:

$$A2 : RxRxR \rightarrow CxC.$$

Subliniem încă odată că orice algoritm se încheie după un număr finit de pași, caracterul de finititudine constituind una dintre proprietățile esențiale ale unui algoritm. Alte caracte-

ristici fundamentale ale unui algoritm sînt: **univocitatea** - fiecare operaţie (succesiunea de operaţii) este în mod unic definită de rezultatele parţiale obţinute prin operaţiile anterioare şi **generalitatea** - algoritmul rezolvă o problemă pentru orice element al mulţimii datelor de intrare corespunzătoare, nu numai pentru anumite date de intrare particulare.

Noţiunea de algoritm ocupă un loc central în informatică. Intr-adevăr, rezolvarea la calculator a oricărei probleme, indiferent de natura ei, presupune algoritmizarea prealabilă a acesteia, adică elaborarea unui algoritm care rezolvă problema respectivă, determină în mod efectiv soluţia problemei.

Un rol important îl are latura algoritmică asupra înţelegerii problemelor de matematică. Calculul algoritmic forţează precizia gîndirii, algoritmizarea unei metode matematice presupune detalierea tuturor etapelor ei, urmărirea pas cu pas a fiecăreia dintre aceste etape, deci o adîncă pătrundere a întregii metode.

2.2. Descrierea algoritmilor.

Odată cu elaborarea unui algoritm, acesta trebuie prezentat sub o formă cît mai precisă şi clară, pentru a putea fi transpus, în continuare, sub forma unui program acceptat de calculator. Această operaţie constituie activitatea de descriere a algoritmilor.

Avînd în vedere scopul propus, considerăm potrivit să folosim pentru descrierea algoritmilor un pseudolimbaj şi limbajul schemelor logice. Primul foloseşte simboluri şi propoziţii din matematică şi se bucură de o largă accesibilitate, iar al doilea - deja cu mare grad de familiaritate - se caracterizează prin expresivitate.

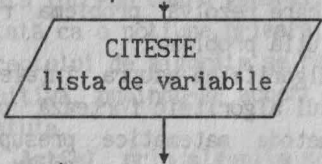
Atît în pseudolimbaj cît şi în limbajul schemelor logice se va folosi, pe lîngă simbolurile matematice uzuale şi simbolul de atribuire ":=", a cărui semnificaţie constă în a atribui variabilei din partea stîngă valoarea expresiei din partea dreaptă. De exemplu, $x:=10$ înseamnă că x primeşte valoarea 10, iar $i:=i+1$ reprezintă mărirea valorii variabilei "i" cu o unitate. Se obişnuieşte să se spună că lui x i se atribuie valoarea 10 respectiv lui i i se atribuie $i+1$.

Schema logică poate fi interpretată ca o reprezentare grafică a algoritmului, formată din blocuri şi segmente (arce) orientate, blocurile marcînd operaţiile ce trebuie efectuate, iar ar-

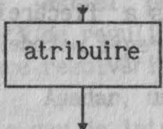
cele indică relația de succesiune a operațiilor. Blocurile folosite de limbajul schemelor logice sînt de următoarele tipuri: bloc de calcul, bloc de decizie, bloc de intrare/ieșire, bloc delimitator, etc. Pentru marcarea acestor blocuri convenim să folosim următoarele simboluri:



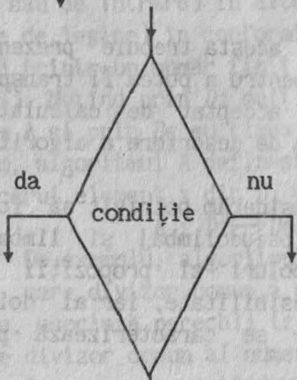
pentru marcarea începutului schemei logice.



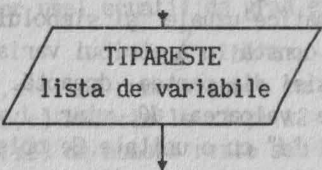
pentru marcarea datelor de intrare: lista de variabile conține variabilele ce constituie datele inițiale ale problemei.



pentru marcarea atribuirilor



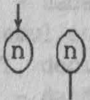
pentru marcarea etapelor de decizie: în cazul în care condiția este verificată se urmează ramura marcată cu "da", iar în caz contrar se urmează ramura marcată cu "nu".



pentru marcarea datelor de ieșire: conține variabilele ce reprezintă rezultatele finale.



pentru marcarea sfîrșitului schemei logice.



pentru racordarea unei scheme logice: se foloseste, de obicei, în cazul în care o schemă logică se scrie pe mai multe pagini.

2.3. Exemple.

În continuare vom da mai multe exemple de algoritmi descriși în cele două limbaje considerate.

Spre deosebire de simbolurile matematice obișnuite se va folosi simbolul "*" pentru marcarea operației de înmulțire și simbolul "↑" pentru indicarea operației de ridicare la putere, simboluri care vor fi folosite și în capitolele următoare.

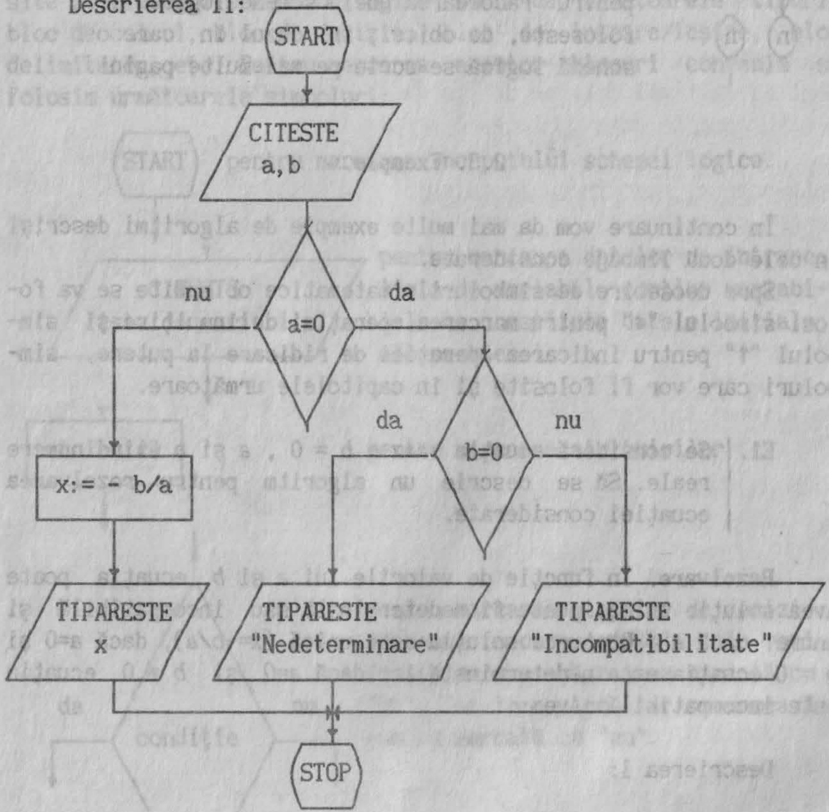
E1. Se consideră ecuația $a*x + b = 0$, a și b fiind numere reale. Să se descrie un algoritm pentru rezolvarea ecuației considerate.

Rezolvare. În funcție de valorile lui a și b, ecuația poate avea soluție unică, poate fi nedeterminată sau incompatibilă și anume: dacă $a \neq 0$ atunci soluția este unică ($x=-b/a$), dacă $a=0$ și $b = 0$ ecuația este nedeterminată iar dacă $a=0$ și $b \neq 0$ ecuația este incompatibilă. Avem:

Descrierea I:

1. Date inițiale: a,b - numere reale;
2.1. Dacă a ≠ 0 atunci x:=-b/a și se trece la 3.1.
2.2. Dacă a = 0 atunci
 Dacă b = 0 atunci se trece la 3.2.
 altfel se trece la 3.3.
3. Rezultate :
 3.1. Numărul x;
 3.2. Mesajul "Nedeterminare";
 3.3. Mesajul "Incompatibilitate";

Descrierea II:



E2. Să se descrie un algoritm pentru rezolvarea sistemului:

$$a*x + b*y = c$$
$$d*x + e*y = f$$

unde a,b,c,d,e,f sînt numere reale.

Rezolvare. Se știe că sistemul poate fi compatibil determinat, compatibil nedeterminat sau incompatibil, în funcție de valorile coeficienților și anume:

- dacă $a/d \neq b/e$ deci $a*e - b*d \neq 0$ atunci sistemul are soluția unică:

$$x = \frac{c*e - b*f}{a*e - b*d}, \quad y = \frac{a*f - c*d}{a*e - b*d}.$$

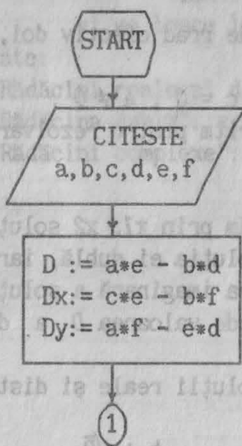
- dacă $a*e-b*d=0$ şi $a*f-c*d=0$ (cînd vom avea şi $c*e-b*f=0$) sistemul este compatibil nedeterminat. Într-adevăr, avem $a/d=b/e=c/f$, deci sistemul se reduce la o singură ecuaţie;

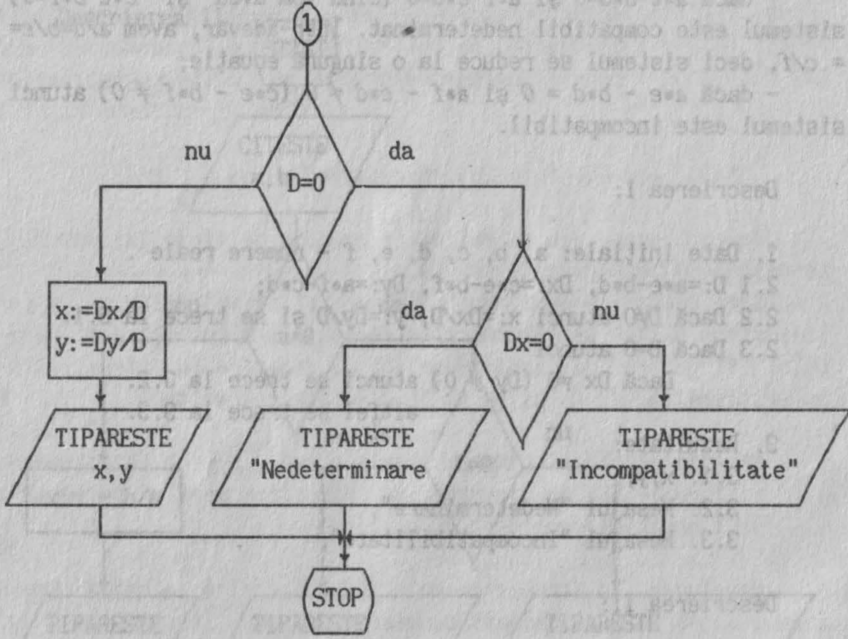
- dacă $a*e - b*d = 0$ şi $a*f - c*d \neq 0$ ($c*e - b*f \neq 0$) atunci sistemul este incompatibil.

Descrierea I:

1. Date iniţiale: a, b, c, d, e, f - numere reale .
2.1 D:=a*e-b*d, Dx:=c*e-b*f, Dy:=a*f-c*d;
2.2 Dacă D≠0 atunci x:=Dx/D, y:=Dy/D şi se trece la 3.1.
2.3 Dacă D=0 atunci
    Dacă Dx ≠0 (Dy ≠ 0) atunci se trece la 3.2.
    altfel se trece la 3.3.
3. Rezultate:
    3.1. x,y;
    3.2. Mesajul "Nedeterminare";
    3.3. Mesajul "Incompatibilitate";

Descrierea II:





E3 | Se consideră ecuația de grad efectiv doi, cu coeficicienți reali:

$$a*x\uparrow 2 + b*x + c = 0 \ , \ a \neq 0 \ .$$

Să se descrie un algoritm pentru rezolvarea ecuației date .

Rezolvare. Convenim să notăm prin *x1*, *x2* soluțiile reale și diferite ale ecuației, prin *x* soluția ei dublă, iar prin *Rex* și *Imx* partea reală respectiv partea imaginară a soluțiilor complexe. Natura soluțiilor este dată de valoarea *D* a discrimnantului ecuației, $D=b\uparrow 2-4*a*c$, astfel:

- dacă $D > 0$ ecuația are soluții reale și distincte:

$$x1 = \frac{-b - \sqrt{D}}{2 * a}, \quad x2 = \frac{-b + \sqrt{D}}{2 * a} \ .$$

- dacă $D = 0$ soluțiile sînt reale și egale, deci soluția dublă este $x = - b/(2*a)$;

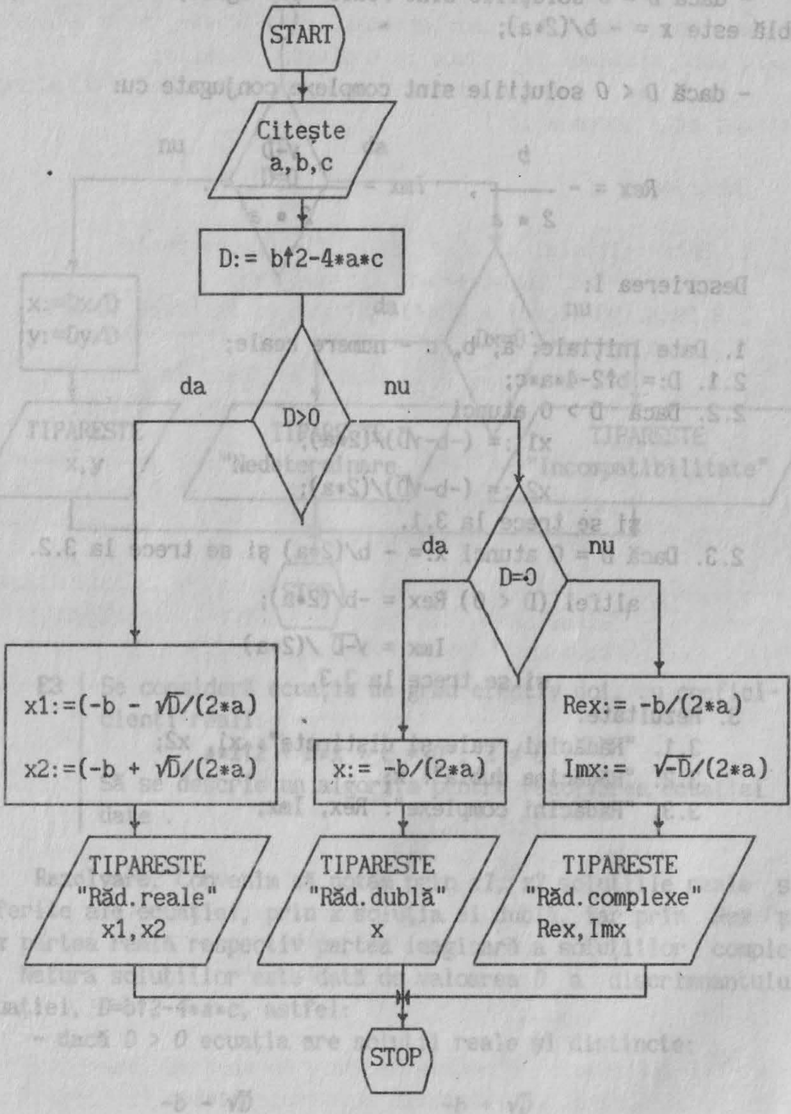
- dacă $D < 0$ soluțiile sînt complexe conjugate cu:

$$Rex = - \frac{b}{2 * a}, \quad Imx = \frac{\sqrt{-D}}{2 * a}.$$

Descrierea I:

1. Date inițiale: a, b, c - numere reale;
2.1. D:= b↑2-4*a*c;
2.2. Dacă D > 0 atunci
x1 := (-b-√D)/(2*a);
x2 := (-b-√D)/(2*a);
și se trece la 3.1.
2.3. Dacă D = 0 atunci x:= - b/(2*a) și se trece la 3.2.
altfel (D < 0) Rex = -b/(2*a);
Imx = √-D /(2*a)
și se trece la 3.3.
3. Rezultate:
3.1. "Rădăcini reale și distincte": x1, x2;
3.2. "Rădăcina dublă": x;
3.3. "Rădăcini complexe": Rex, Imx;

Descrierea II:



E4. Fie *n* un număr natural, *n>1* și a(1),...,a(*n*) numere reale. Să se descrie un algoritm pentru calculul mediei aritmetice a numerelor *a*(*1*),...,*a*(*n*).

Rezolvare. Fie *MA* media cerută; $MA = (a(1) + \ldots + a(n))/n$. Folosind proprietatea de asociativitate a operaţiei de adunare, calculul sumei se poate efectua în modul următor; $MA := a(1)$, $MA := MA + a(2), \ldots, MA := MA + a(n)$ sau $MA := a(1)$, $MA := MA + a(i), i = 2, \ldots, n$.

Observaţie: Pe viitor convenim să notăm prin $(a(i), i=1, n)$ elementele $a(1), \ldots, a(n)$.

Descrierea I:

1. Date iniţiale: n-număr natural:
   (a(i),i=1,n)-numere reale;

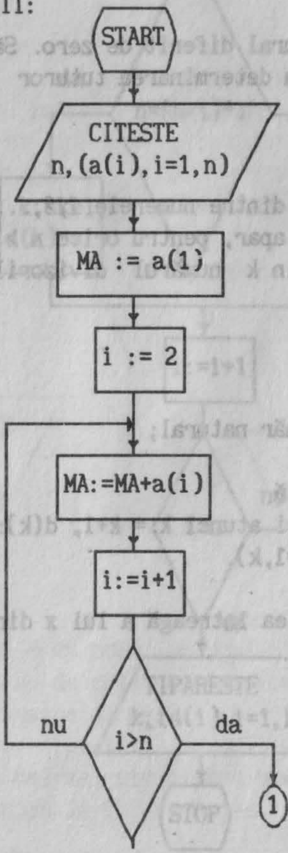
2.1. MA:= a(1);

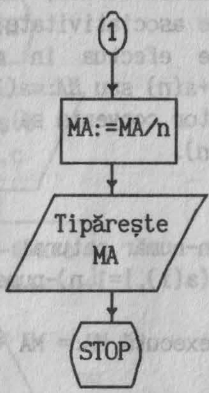
2.2. Pentru i=2,n execută MA:= MA + a(i);

2.3. MA:= MA/n;

3. Rezultate: MA;

Descrierea II:





E5. | Fie n un număr natural diferit de zero. Să se descrie un algoritm pentru determinarea tuturor divizorilor lui *n*.

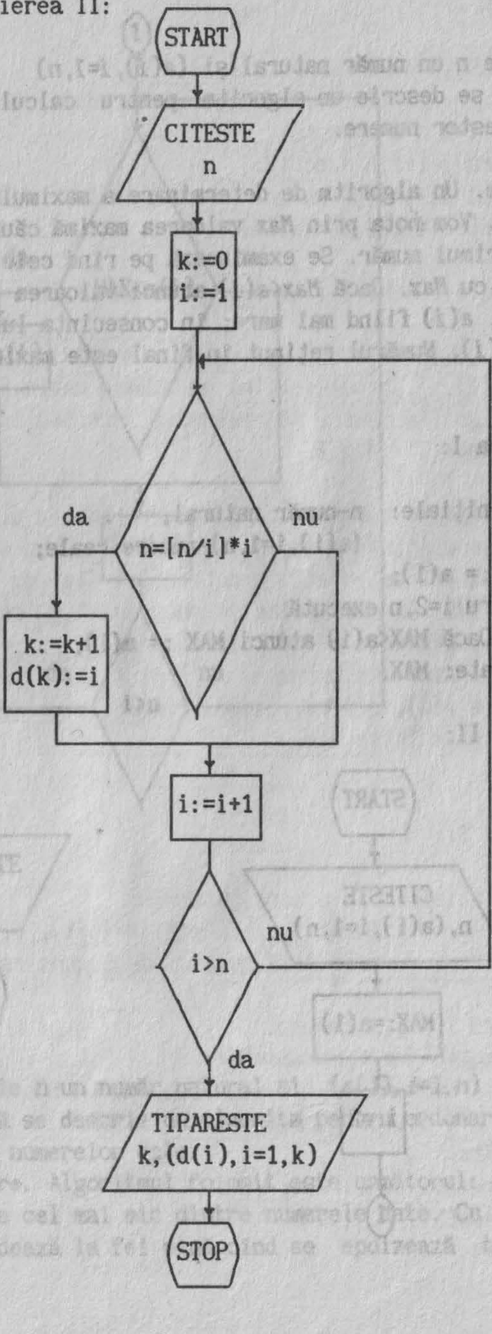
Rezolvare. Se selectează dintre numerele *1,2,...,n* cele care divid pe *n*. Printre divizori apar, pentru orice *n* > 1, divizorii improprii *1* și *n*. Să notăm prin *k* numărul divizorilor și prin (*d*(*i*), *i*=*1*, *k*) acești divizori.

Descrierea I:

1. Date inițiale: n - număr natural;
2.1. k := 0
2.2. Pentru i= 1,n execută
   Dacă n = [n/i]*i atunci k:= k+1, d(k):= i;
3. Rezultate: k, (d(i),i=1,k).

Observatie. [*x*] este partea întreagă a lui *x* din R.

Descrierea II:



E6. Fie n un număr natural și $(a(i), i=1,n)$ numere reale. Să se descrie un algoritm pentru calculul maximului acestor numere.

Rezolvare. Un algoritm de determinare a maximului este dat în continuare. Vom nota prin *Max* valoarea maximă căutată. Se ia *Max* egal cu primul număr. Se examinează pe rînd celelalte numere, comparîndu-le cu *Max*. Dacă *Max*<*a*(*i*) atunci valoarea lui *Max* nu este cea bună, *a*(*i*) fiind mai mare; în consecința lui *Max* i se va da valoarea *a*(*i*). Numărul reținut în final este maximul numerelor date.

Descrierea I:

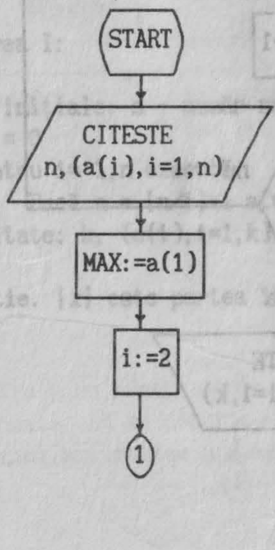
1. Date inițiale: n-număr natural,
   (a(i),i=1,n)-numere reale;

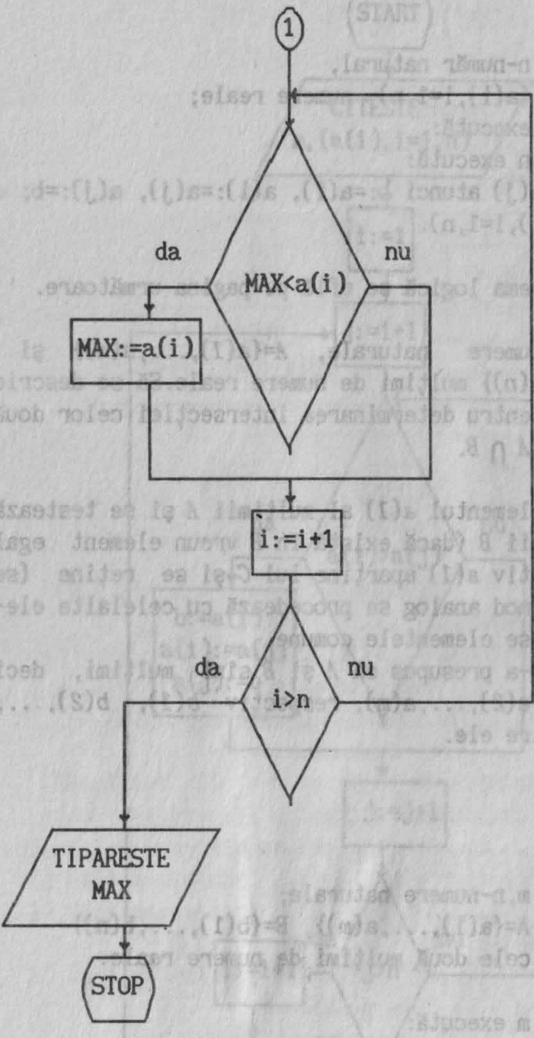
2.1. MAX := a(1);

2.2. Pentru i=2,n execută
     Dacă MAX<a(i) atunci MAX := a(i);

3. Rezultate: MAX.

Descrierea II:





E7. Fie *n* un număr natural și $(a(i), i=1,n)$ numere reale. Să se descrie un algoritm pentru ordonarea crescătoare a numerelor date.

Rezolvare. Algoritmul folosit este următorul: se aduce pe prima poziție cel mai mic dintre numerele date. Cu restul numerelor se procedează la fel pînă cînd se epuizează toate numerele considerate.

Descrierea I:

1. Date iniţiale: n-număr natural,
   (a(i),i=1,n)- numere reale;
2. Pentru i=1,n-1 execută:
   Pentru j=i+1,n execută:
   Dacă a(i)>a(j) atunci b:=a(i), a(i):=a(j), a(j):=b;
3. Rezultate: (a(i),i=1,n).

Descrierea II: Schema logică se află pe pagina următoare.

E8. Fie *m*,*n* - numere naturale, *A*={*a*(*1*),...,a(*m*)} şi *B*={*b*(*1*),...,*b*(*n*)} mulţimi de numere reale. Să se descrie un algoritm pentru determinarea intersecţiei celor două mulţimi: $C = A \cap B$.

Rezolvare. Se ia elementul *a*(*1*) al mulţimii *A* şi se testează dacă el aparţine mulţimii *B* (dacă există în *B* vreun element egal cu *a*(*1*)). În caz afirmativ *a*(*1*) aparţine lui *C* şi se reţine (se notează prin *c*(*1*)). În mod analog se procedează cu celelalte elemente din *A*, reţinîndu-se elementele comune.

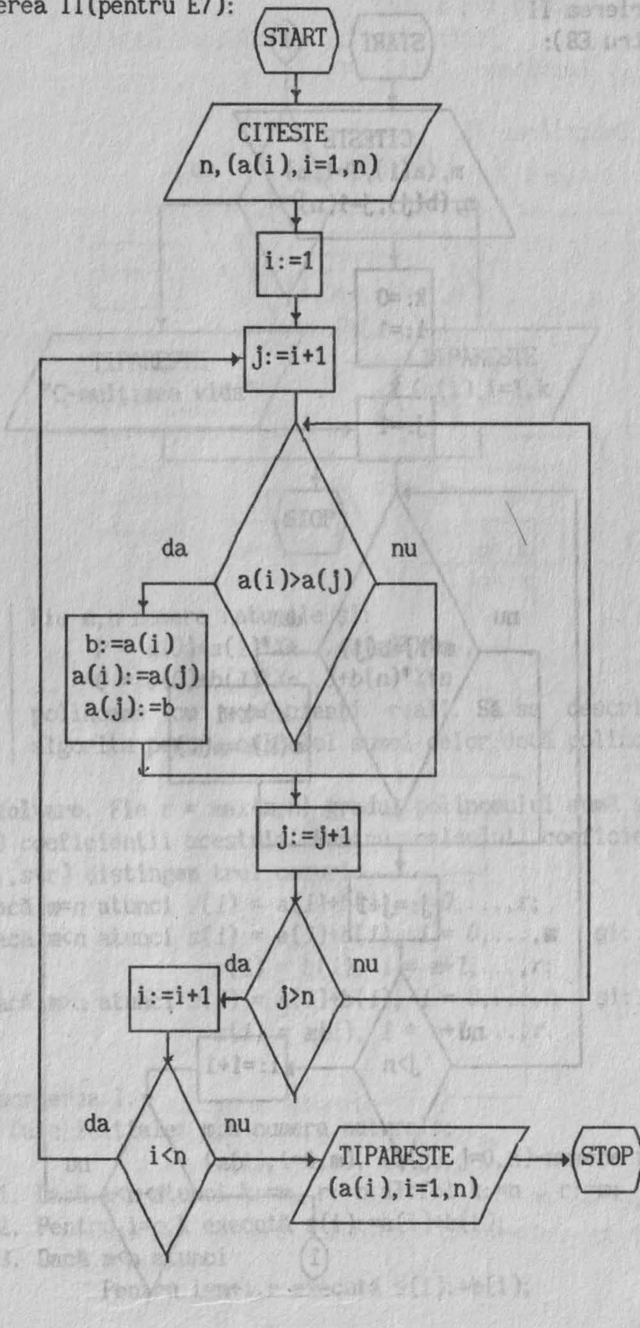
Observaţie. Aici s-a presupus că *A* şi *B* sînt mulţimi, deci toate elementele *a*(*1*), *a*(*2*),...,*a*(*m*), respectiv *b*(*1*), *b*(*2*),..., *b*(*n*) sînt distincte între ele.

Descrierea I:

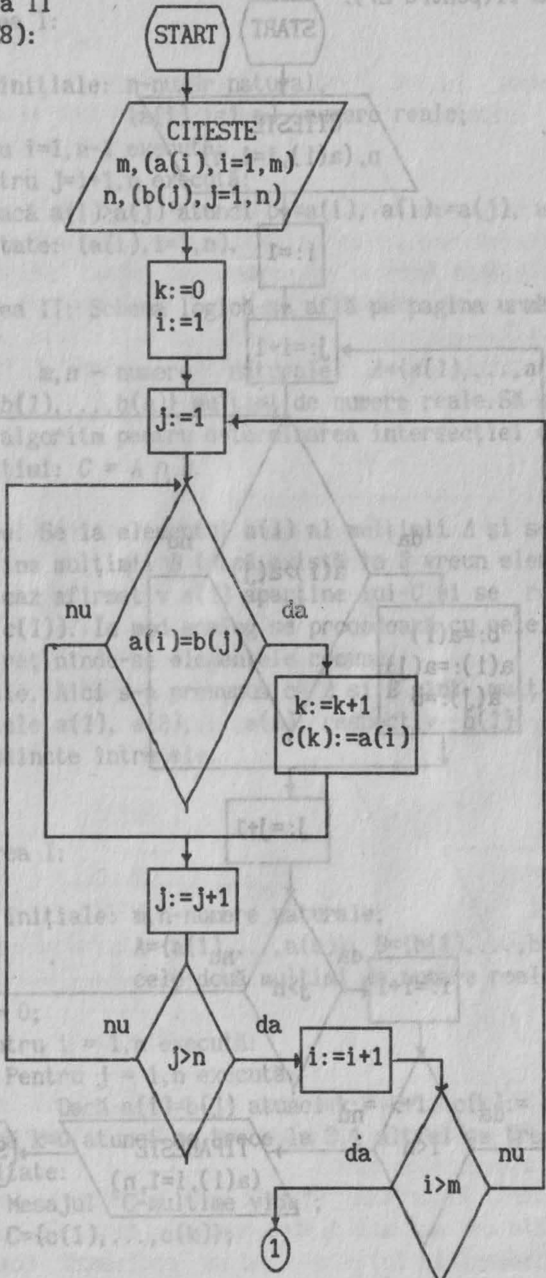
1. Date iniţiale: m,n-numere naturale;
   A={a(1),...,a(m)}, B={b(1),...,b(n)}-
   cele două mulţimi de numere reale.

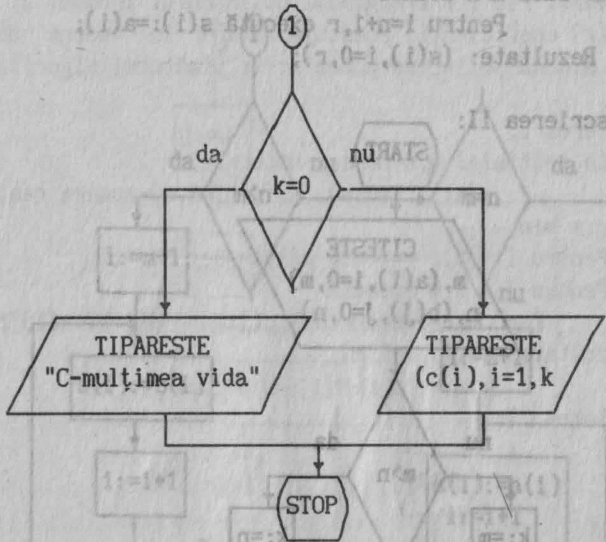
2.1. k:= 0;
2.2. Pentru i = 1,m execută:
   Pentru j = 1,n execută:
   Dacă a(i)=b(j) atunci k:= k+1, c(k):= a(i);
2.3. Dacă k=0 atunci se trece la 3.1 altfel se trece la 3.2.
3. Rezultate:
   3.1. Mesajul "C-multime vida";
   3.2. C={c(1),...,c(k)};

Descrierea II(pentru E7):



Descrierea II
(pentru E8):





E9. Fie m,n numere naturale şi:

$P = a(0)+a(1)*X+...+a(m)*X\uparrow m$,

$Q = b(0)+b(1)*X+...+b(n)*X\uparrow n$

polinoame cu coeficienţi reali. Să se descrie un algoritm pentru calculul sumei celor două polinoame.

Rezolvare. Fie $r = \max\{m,n\}$ gradul polinomului sumă şi $s(0)$, ..., $s(r)$ coeficienţii acestuia. Pentru calculul coeficienţilor $s(0),...,s(r)$ distingem trei cazuri:

1) dacă $m=n$ atunci $s(i) = a(i)+b(i)$, $i=0,...,r$;
2) dacă $m<n$ atunci $s(i) = a(i)+b(i)$, $i = 0,...,m$ şi:
$s(i) = b(i)$, $i = m+1,...,r$;
3) dacă $m>n$ atunci $s(i) = a(i)+b(i)$, $i = 0,...,n$ şi:
$s(i) = a(i)$, $i = n+1,...,r$.

Descrierea I:

1. Date iniţiale: m,n-numere naturale;
   (a(i),i=0,m), (b(j),j=0,n)-numere reale;

2.1. Dacă m<n atunci k:=m, r:=n altfel k:=n , r:=m;

2.2. Pentru i=o,k execută s(i):=a(i)+b(i);

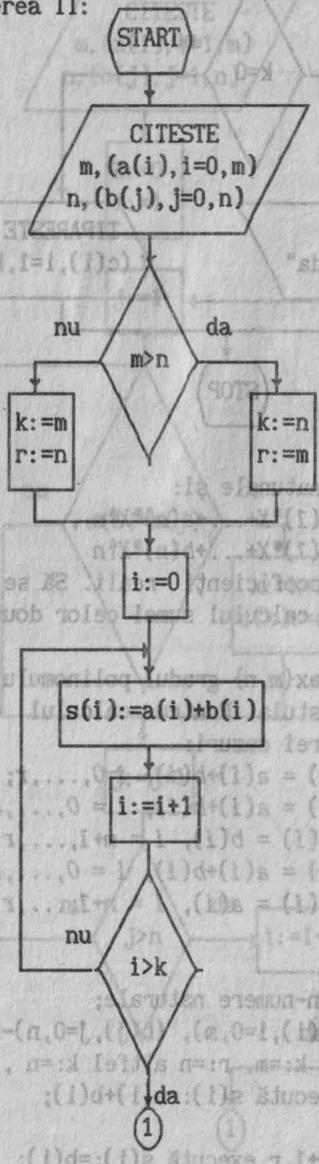
2.3. Dacă m<n atunci
    Pentru i=m+1,r execută s(i):=b(i);

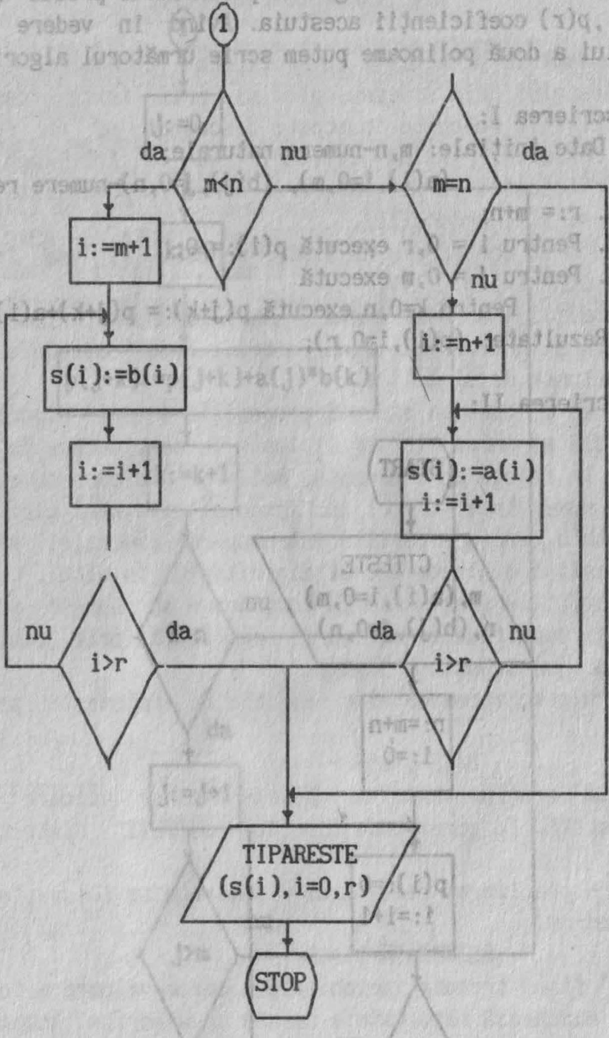
2.4. Dacă m>n atunci

Pentru i=n+1,r execută s(i):=a(i);

3. Rezultate: (s(i),i=0,r);

Descrierea II:





E10. Fie m,n numere naturale şi:

$$P = a(0)+a(1)*X+\ldots+a(m)*X\uparrow m,$$
$$Q = b(0)+b(1)*X+\ldots+b(n)*X\uparrow n$$

polinoame cu coeficienţi reali. Să se descrie un algoritm pentru calculul produsului celor două polinoame.

Rezolvare. Fie *r* = *m+n* gradul polinomului produs și *p(0)*, *p(1)*,...,*p(r)* coeficienții acestuia. Avînd în vedere definiția produsului a două polinoame putem scrie următorul algoritm:

Descrierea I:

1. Date inițiale: m,n-numere naturale;
   (a(i),i=0,m), (b(j),j=0,n)-numere reale;

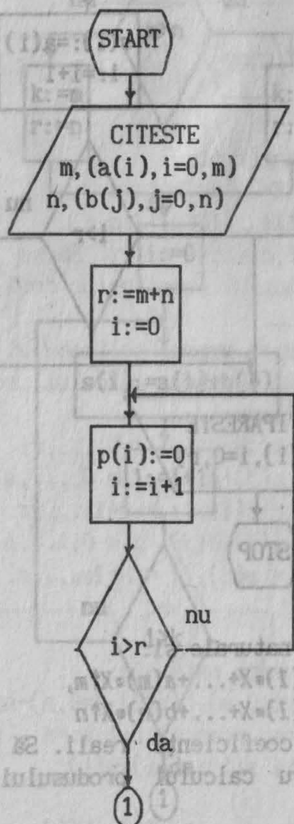
2.1. r:= m+n;

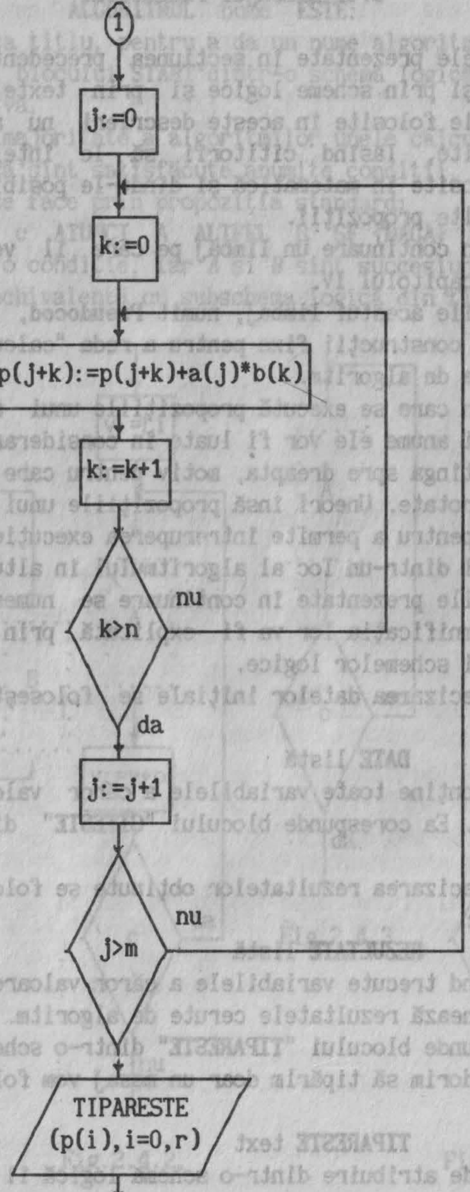
2.2. Pentru i = 0,r execută p(i):= 0;

2.3. Pentru j = 0,m execută
   Pentru k=0,n execută p(j+k):= p(j+k)+a(i)*b(j);

3. Rezultate: (p(i),i=0,r);

Descrierea II:




1
j:=0
k:=0
p(j+k):=p(j+k)+a(j)*b(k)
k:=k+1
k>n
nu
da
j:=j+1
j>m
nu
TIPARESTE
(p(i),i=0,r)
STOP

2.4. Limbajul Pseudocod.

În exemplele prezentate în secţiunea precedentă algoritmii au fost descrişi prin scheme logice şi prin texte (descrierile I). Propoziţiile folosite în aceste descrieri nu au fost însă riguros definite, lăsînd cititorii să le înţeleagă conform notaţiilor folosite în matematică şi dîndu-le posibilitatea să-şi imagineze şi alte propoziţii.

Definim în continuare un limbaj pe care îl vom folosi în exemplele din capitolul IV.

Propoziţiile acestui limbaj, numit **Pseudocod**, folosesc cuvinte cheie şi construcţii fixe pentru a reda "calculele elementare" efectuate de algoritm.

Ordinea în care se execută propoziţiile unui algoritm este cea naturală şi anume ele vor fi luate în considerare de sus în jos şi de la stînga spre dreapta, motiv pentru care nu este nevoie să fie numerotate. Uneori însă propoziţiile unui algoritm vor fi numerotate pentru a permite întreruperea execuţiei secvenţiale, deci saltul dintr-un loc al algoritmului în altul.

Propoziţiile prezentate în continuare se numesc propoziţii standard şi semnificaţia lor va fi explicată prin echivalentul lor în limbajul schemelor logice.

Pentru precizarea datelor iniţiale se foloseşte propoziţia standard:

DATE listă

unde "listă" conţine toate variabilele a căror valoare iniţială este cunoscută. Ea corespunde blocului "**CITESTE**" dintr-o schema logică.

Pentru precizarea rezultatelor obţinute se foloseşte propoziţia standard:

REZULTATE listă

în "listă" fiind trecute variabilele a căror valoare a fost obţinută; ele marchează rezultatele cerute de algoritm. Această propoziţie corespunde blocului "**TIPARESTE**" dintr-o schemă logică. In cazul în care dorim să tipărim doar un mesaj vom folosi propoziţia:

TIPARESTE text

Blocului de atribuire dintr-o schemă logică îi corespunde în Pseudocod propozitia standard:

FIE v := expresie

cuvintul **FIE** fiind opţional.

Propoziţia:

ALGORITMUL nume **ESTE**:

se foloseşte ca titlu, pentru a da un nume algoritmului descris. Ea joacă rolul blocului START dintr-o schemă logică. Prezenţa ei este facultativă.

In marea majoritate a algoritmilor unele calcule se efectuează numai dacă sînt satisfăcute anumite condiţii. Redarea acestor situaţii se face prin propoziţia standard:

DACA c **ATUNCI** A **ALTFEL** B **SF {DACA}**

unde "*c*" este o condiţie, iar *A* şi *B* sînt succesiuni de propoziţii. Ea este echivalentă cu subschema logică din fig.2.4.1.

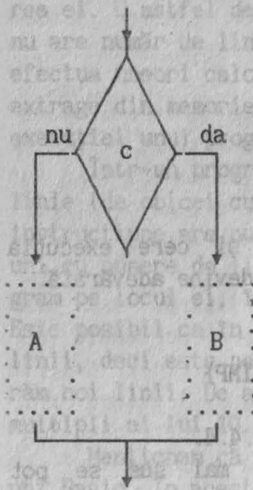


Fig.2.4.1.

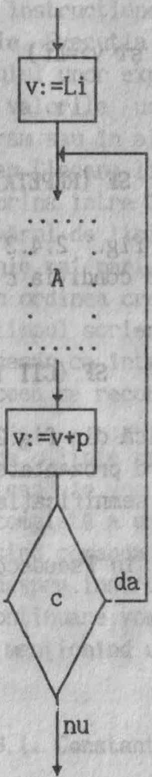


Fig.2.4.2.



Fig.2.4.3.

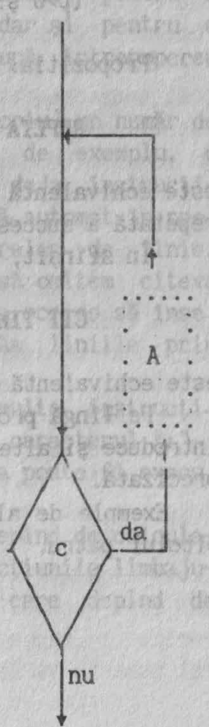


Fig.2.4.4.

În cazul cînd *B* este vidă se foloseşte varianta simplificată

DACA c **ATUNCI** A **SF {DACA}**

Adesea este nevoie de repetarea unor calcule pentru valori bine precizate ale unei variabile de control. O astfel de repetare se redă prin propoziția standard:

PENTRU v = Li, Lf [,p] EXECUTA

A

SF {PENTRU}

care este echivalentă cu subschema logică din fig.2.4.2, unde conditia *c* este:

(p>0 și v<=Lf) sau (p<0 și v>=Lf) .

Propozitia:

REPETA A PINA CIND c SF {REPETA}

este echivalentă cu subschema din fig. 2.4.3 și cere execuția repetată a succesiunii *A* pînă cînd condiția *c* devine adevărată.

În sfîrșit, propoziția:

CIT TIMP c EXECUTA A SF {CIT TIMP}

este echivalentă cu subschema logică din fig.2.4.4.

Pe lîngă propozițiile standard prezentate mai sus se pot introduce și alte propoziții, dar semnificația lor trebuie bine precizată.

Exemple de algoritmi descriși în Pseudocod se găsesc în capitolul patru.

3. LIMBAJUL BASIC.

Limbajul BASIC a fost elaborat de un colectiv de la colegiul Dartmouth (SUA) în anul 1965. Numele limbajului este format din inițialele cuvintelor din următoarea frază: Beginner's All-purpose Symbolic Instruction Code (cod simbolic de instrucțiuni, de scop general, pentru începători).

În continuare vom prezenta instrucțiunile limbajului Basic. Trebuie să menționăm că există două moduri de execuție a instrucțiunilor Basic: imediată și într-un program. La modul de execuție imediată o instrucțiune se execută îndată ce s-a terminat scrierea ei. O astfel de instrucțiune se recunoaște prin faptul că ea nu are număr de linie. Execuția imediată se folosește pentru a efectua uneori calculul unor expresii numerice, dar și pentru a extrage din memorie valorile unor variabile după întreruperea execuției unui program sau în alte scopuri.

Într-un program fiecare linie are (obligatoriu) un număr de linie (de obicei cuprins între 1 și 9999). Dacă, de exemplu, o instrucțiune are numărul de linie 45 și am scris deja instrucțiuni cu numere de linie mai mari, ea va fi plasată automat în program pe locul ei, în ordinea crescătoare a numerelor de linie. Este posibil ca în timpul scrierii unui program să omitem cîteva linii, deci este necesar ca între două linii deja scrise să inserăm noi linii. De aceea se recomandă să numerotăm liniile prin multipli ai lui 10.

Menționam că pe o linie pot fi scrise mai multe instrucțiuni Basic. În acest caz ele vor fi separate prin caracterul ":".

După scrierea completă a unui program acesta poate fi executat la cerere, folosind comanda RUN.

Unele detalii despre instrucțiunile Basic depind de calculatorul folosit. În continuare vom prezenta instrucțiunile limbajului Basic standard, menționînd uneori aspectele care depind de calculator.

3.1. Constante si variabile.

În scrierea instrucțiunilor intervin două tipuri de mărimi: constante, care pot fi numerice sau șiruri de caractere și variabile.

Constantele numerice sînt șiruri de cifre, precedate sau nu

de semn (+ sau -), cu sau fără punctul zecimal, care reprezintă numerele întregi sau reale întîlnite în matematică. Constantele numerice se pot exprima:

- în format întreg: 235 ; -39 ; +5238 ;
- în format real: 3.1415 ; -.864; +124.678 (se observă că pentru separarea părţii întregi de partea fracţionară s-a folosit punctul în loc de virgulă);
- în format exponenţial: 234.5E+13 ; .2531E-14. Prin *mEe* se înţelege numărul obţinut prin înmulţirea lui *m* cu 10 la puterea *e*. Se poate folosi orice constantă numerică ce aparţine intervalului [1.7E-38, 1.7E+38], dar acest interval poate să difere în funcţie de calculatorul utilizat.

Constantele sir de caractere sînt şiruri de caractere scrise între ghilimele ("). Exemple: "eroare"; "Nu exista solutie" ; "Universitatea din Cluj-Napoca".

Mărimea a cărei valoare se poate schimba pe parcursul execuţiei unui program se numeşte **variabilă**. Ea se reprezintă printr-un identificator, care este numele variabilei şi are rezervat un loc în memoria calculatorului, loc în care se poate depune o singură valoare numerică. Deci variabila are în fiecare moment o singură valoare; ea se mai numeşte şi **variabilă simplă**.

Identificatorul Basic este format dintr-o literă, sau dintr-o literă urmată de o cifră. Multe calculatoare personale acceptă însă şi identificatori de lungime mai mare decît doi. În acest caz un identificator este format din litere şi cifre, primul caracter fiind obligatoriu o literă. Sînt acceptate atît litere mari cît şi litere mici, dar o literă mică este identică cu cea mare.

Pentru reprezentarea şirurilor de numere sau a matricelor se folosesc tablourile cu unul sau mai mulţi indici. Fiecărui tablou i se asociază un identificator ce reprezintă numele tabloului. Toate elementele tabloului au acelaşi nume - numele tabloului, dar sînt deosebite între ele prin indici, care pot fi constante, variabile, sau expresii aritmetice. Un element al tabloului se mai numeşte **variabilă cu indici**.

3.2. Instructiunea DIM.

Pentru fiecare tablou trebuie să rezervăm o zonă de memorie unde se vor depune valorile variabilelor cu indici care formează tabloul. Mărimea acestei zone de memorie trebuie precizată de

programator prin instrucţiunea DIM. Aceasta are sintaxa:

n DIM *tablou*

unde "*tablou*" are sintaxa

id (nr {,nr})

Prin {*text*} se indică faptul că textul dintre acolade se poate repeta ori de cîte ori, inclusiv poate să lipsească. În faţa instrucţiunii, *n* reprezintă numărul liniei. Exemplu:

22 DIM A(20,12)

Instrucţiunea **DIM** are ca efect rezervarea unei zone de memorie formată din *(n1+1)*(n2+1)*...*(nk+1)* locaţii pentru un tablou care are sintaxa id(n1, n2, ... , nk). Pe timpul execuţiei programului valoarea primului indice trebuie să fie cuprinsă între 0 şi n1, ... , valoarea indicelui de rang k - între 0 şi nk. Unele calculatoare (de exemplu HC-85) nu acceptă indice egal cu 0, deci şi mărimea zonei alocate tabloului diferă. Există variante ale limbajului Basic care acceptă mai multe declaraţii într-o instrucţiune DIM. Un tablou unidimensional (cu un singur indice) se mai numeşte **vector**, iar unul bidimensional, **matrice**.

3.3. Functii Basic.

Pentru calculul valorii unor funcţii programatorul are posibilitatea să şi le definească, aşa cum se va arăta mai tîrziu. Există însă funcţii definite deja, pe care le poate utiliza direct, respectînd regulile de folosire a acestora. Aceste funcţii definite implicit sînt:

ABS(X) pentru a calcula |X| ;
SGN(X) pentru a calcula **signum(X)** (semnul lui X);
INT(X) pentru a calcula [X] (partea întreagă a lui X) ;
COS(X) pentru a calcula **cos(X)** ⎫
SIN(X) pentru a calcula **sin(X)** ⎬ pentru X în radiani
TAN(X) pentru a calcula **tg(X)** ⎭
ATN(X) pentru a calcula **arctg(X)**, rezultatul în radiani;
SQR(X) pentru a calcula radical din X, pentru X>=0;
EXP(X) pentru a calcula e la puterea X;
LOG(X) pentru a calcula ln X cînd X >0.

3.4. Comentarii.

Pentru orice program este indicat să se precizeze, sub formă de comentariu, cel puţin problema pe care o rezolvă. In acest scop se foloseşte instrucţiunea **REM** (de la REMark) care nu este luată în considerare de calculator, fiind la dispoziţia programatorilor pentru a plasa comentarii în anumite locuri din program. Ea are forma:

n **REM** *comentariu*

n fiind numărul de linie. Pe lîngă problema pe care o rezolvă programul respectiv, este bine ca să se specifice variabilele folosite şi semnificaţia lor.

3.5. Expresii aritmetice. Instructiunea de atribuire.

Instrucţiunea de atribuire are forma:

n **LET** *v = expresie*

unde *v* este numele unei variabile (cu sau fără indici), iar *expresie* este o expresie aritmetică.

O expresie aritmetică corespunde unei expresii algebrice din matematică dacă este scrisă după regulile limbajului Basic. Astfel, toţi indicii trebuie scrişi în acelaşi rînd cu variabila indexată, din care cauză ei vor fi scrişi între paranteze. Ordinea de efectuare a celor patru operaţii este cea din matematică, iar pentru a preciza o altă ordine se folosesc parantezele, aşa cum se obişnuieşte. Cele patru operaţii cunoscute se notează în Basic prin +, -, *, respectiv /. Există şi operaţia de ridicare la putere, cea mai prioritară, notată prin semnul ↑ folosită pentru a calcula valoarea funcţiei putere "*a la puterea x*", pentru *a* pozitiv. Unele calculatoare admit şi *a* negativ cînd *a*↑*x* are sens.

De exemplu, prin *A(i,j+1)* se notează elementul matricei *A* din linia *i* coloana *j+1*. Prin *W(2*i-1)* se notează componenta vectorului *W* cu indicele *2*i-1*. Cu aceste elemente putem construi expresia aritmetică: *A(i,J+1) * SIN(W(2*i-1))*.

Instrucţiunea de atribuire are ca efect calculul expresiei din partea dreaptă a semnului egal şi atribuirea valorii obţinute variabilei *v*. Deci variabila *v* va primi (pe locul din memoria

calculatorului rezervat acestei variabile) valoarea obținută din calculul expresiei; vechea valoare a variabilei *v*, dacă *v* a avut o valoare, dispare.

3.6. Instrucțiuni de intrare/ieșire.

Pentru extragerea rezultatelor din memoria calculatorului se folosește instrucțiunea PRINT, care are sintaxa:

n **PRINT** *listaprint*

unde "*listaprint*" este formată din elemente permise separate între ele prin separatorii "," sau ";". Elementele permise în *listaprint* sînt variabilele (simple sau cu indici) și constantele (numerice, sau de tip șir de caractere pentru tipărirea unor mesaje). Prin instrucțiunea **PRINT** se cere extragerea din memorie a valorilor variabilelor prezente în *listaprint* și tipărirea lor pe ecran, în timp ce mesajul "*text*" format din textul scris între ghilimele este tipărit nemodificat. O linie pe ecran se împarte în două zone (de obicei cu lungimea de 14 caractere/zonă). Tipărirea se continuă pe același rînd fără nici un spațiu dacă separatorul folosit în fața elementului este ";" și se face în zona următoare dacă separatorul folosit este ",".

Elementele din *listaprint* mai pot fi și expresii aritmetice. În cazul cînd o expresie aritmetică apare în *listaprint*, se cere să se calculeze valoarea acestei expresii și să se tipărească valoarea calculată. Valoarea calculată nu este însă păstrată în memorie.

Așa cum s-a arătat, pentru extragerea rezultatelor sau a unor rezultate intermediare din memoria calculatorului se folosește instrucțiunea **PRINT**. Este însă necesară și introducerea datelor inițiale ale problemei în memoria calculatorului. În acest scop se folosește instrucțiunea **INPUT** care are următoarea sintaxă

n **INPUT** *listainput*

unde *listainput* este o listă asemănătoare cu *listaprint*, dar cu semnificație diferită. Elementele permise în *listainput* sînt constante șir de caractere și variabile cu sau fără indici. Si în acest caz mesajele scrise între ghilimele, prezente în *listainput*, vor fi tipărite pe ecran, dar pentru fiecare variabilă prezentă

în listă, în timpul execuţiei programului calculatorul aşteaptă o valoare numerică, valoare care este introdusă în locaţia de memorie alocată variabilei în cauză. Tocmai pentru a semnala care este această variabilă se folosesc mesajele din *listainput*. Exemple se pot vedea în programele care urmează.

Observatie. Dacă în *listainput* o variabilă apare între paranteze, valoarea ei nu se citeşte ci se tipăreşte.

3.7. Instructiunile STOP si END.

Instrucţiunea STOP opreşte temporar execuţia programului. Execuţia unui program poate fi reluată din punctul în care a fost întrerupt cu ajutorul unei comenzi CONTINUE. Unele implementări Basic folosesc intrucţiunea END pentru a marca sfirşitul programului, altele nu au această instrucţiune şi programul avansează pînă cînd epuizează toate instrucţiunile.

Ca exemplu, cu instrucţiunile prezentate mai sus vom scrie un program care calculează SIN 5 şi numerele 1+sqr(6) şi sqr(2) +sqr(5). Programul este următorul:

```
10  REM se tipăreşte valoarea SIN 5
11  REM şi două numere reale, sume de radicali
12  REM
20  LET X =3.141592653/36
30  LET Y=SIN(X)
40  PRINT "SIN(";X;")=";Y
50  LET   S1 = 1 + SQR(6)
60  LET   S2 = SQR(2) + SQR(5)
60  PRINT "S1="; S1, "S2="; S2
70  STOP
```

3.8. Instructiuni de control: GOTO si IF. Expresii logice.

Într-un program instrucţiunile se execută secvenţial în ordinea crescătoare a numerelor de linie. Pentru a permite şi o altă ordine de execuţie se folosesc instrucţiunile de control GOTO şi IF.

Instrucţiunea GOTO are sintaxa:

n GOTO n1

şi cere continuarea necondiţionată a execuţiei programului de la linia n1.

Instrucţiunea **IF** are sintaxa:

n IF exp.logica THEN instructiune [ELSE instructiune]

şi cere mai întîi calculul valorii expresiei logice scrisă între cuvintele **IF** şi **THEN**. Dacă expresia este adevarată atunci se execută instrucţiunea scrisă după cuvîntul **THEN**, altfel, dacă **ELSE** lipseşte, se va executa instrucţiunea următoare (cu numărul de linie imediat superior lui *n*). Prin scrierea între paranteze drepte a construcţiei **ELSE** s-a indicat faptul că această construcţie nu este obligatorie în instrucţiune. Dacă însă ea apare, atunci se execută instrucţiunea care urmează după **ELSE** în cazul că expresia logică are valoarea de fals. Nu toate calculatoarele personale acceptă construcţia **ELSE**. Majoritatea variantelor Basic acceptă mai multe instrucţiuni atît pe ramura **THEN** cît şi pe ramura **ELSE**, instrucţiuni care sînt despărţite prin ":".

Printr-o expresie logică înţelegem fie un singur operand logic, fie operanzi logici legaţi între ei prin operatorii logici binari **AND** (conjuncţia) şi **OR** (disjuncţia).

Prin operand logic înţelegem fie o expresie relaţională, fie o expresie logică închisă între paranteze, fie negaţia unei expresii logice, deci construcţia:

NOT (expresie logica)

Expresia relaţională este o construcţie de forma:

ea1 *relatie* **ea2**

unde *ea1* şi *ea2* sînt expresii aritmetice, iar *relatie* este unul dintre următorii operatori de relaţie: "=" (egal), "<" (mai mic), "<=" (mai mic sau egal), ">" (mai mare), ">=" (mai mare sau egal) şi "<>" (neegal), Pentru a evalua o expresie relaţională se calculează mai întîi valorile *v1* şi *v2* ale expresiilor *ea1*, respectiv *ea2*. Dacă între *v1* şi *v2* există relaţia scrisă între cele două expresii atunci expresia relaţională ia valoarea de *adevar*, în caz contrar pe cea de *fals*.

Pentru a evalua o expresie logică, se evaluează mai întîi operanzii logici, apoi se efectuează operaţiile logice **AND** şi după aceea operaţiile logice **OR**.

3.9. Programul 1: EUCLID.

Folosind instrucţiunile descrise pînă aici putem scrie un program care calculează cel mai mare divizor comun *d* a două numere *n1* şi *n2* folosind algoritmul lui Euclid, descris mai jos în Pseudocod.

```
1. Algoritmul lui Euclid este:
2. Date  n1, n2
3. Fie   d:= n1 ;     i:= n2
4. Cît timp  i <> 0 execută  q := [d/i]
                             r := d - q*i
                             d := i
                             i := r
   sf {cît}
5. Rezultate d.
```

Programul Basic este dat în continuare:

```
1   REM     Programul 1: EUCLID
2   REM ***************************
10  REM Algoritmul lui Euclid
11  REM Datele problemei:
12  REM    n1, n2 - numere întregi
13  REM Rezultatele obtinute:
14  REM                  d= (n1,n2)
15  REM ****************************
16  REM
20  INPUT "n1=";n1
25  INPUT "n2=";n2
40  LET d=n1:  LET  i=n2
50  IF  i=0  THEN  GOTO  100
60      LET  q = INT(d/i)
65      LET  r = d - q*i
70      LET  d = i
80      LET  i = r
90  GOTO 50
100 PRINT "Cel mai mare div.comun"
110 PRINT "    al numerelor"
120 PRINT "n1="; n1 ; " n2="; n2
130 PRINT "este d="; d
140 STOP
```

3.10. Instructiunea FOR.

Pentru a programa repetarea unor calcule de mai multe ori se foloseşte instrucţiunea **FOR**. Ea poate fi utilizată numai împreună cu o instrucţiune **NEXT**. Sintaxa acestor instrucţiuni şi modul în care ele pot apare într-un program Basic sînt:

```
n1   FOR contor = li  TO  lf  [ STEP p ]
n2   . . .  }
     . . .  }  A
     . . .  }
n3   NEXT contor
```

unde n1<n2<n3. Aici *contor* este o variabilă simplă al cărei identificator este format dintr-o singură literă, iar *li*, *lf* şi *p* sînt expresii aritmetice. Construcţia **STEP** *p* este opţională. În cazul cînd lipseşte se consideră implicit p = 1.

Semnificaţia grupului de instrucţiuni scris mai sus, deci şi a instrucţiunii **FOR**, este echivalentă cu propoziţia Pseudocod:

Pentru *contor = li, lf* [,*p*] executa *A sf* {pentru}

Deci se atribuie variabilei contor valoarea expresiei *li*. Dacă este verificată condiţia:

$$(lf - contor) * p >= 0 \qquad (3.1)$$

atunci se execută grupul de instrucţiuni notat prin A. Instrucţiunea **NEXT** din linia *n3* cere modificarea variabilei contor, căreia i se adaugă pasul p. Dacă noua valoare a variabilei contor verifică condiţia (3.1) atunci se execută din nou grupul de instrucţiuni A începînd de la linia n2; în caz contrar se trece la linia următoare instrucţiunii NEXT.

În scrierea programelor se recomandă folosirea spaţiilor libere pentru a separa diferite construcţii sintactice şi a mări claritatea scrierii. De aceea spaţiul nu are valoare sintactică şi poate fi folosit în acest scop.

3.11. Programul 2: PROGRESIE.

Ca exemplu, vom scrie în continuare un program Basic care tipăreşte primii n termeni ai unei progresii aritmetice cu raţia *r* şi primul termen egal cu *a*. În program vom nota prin variabila *t* un termen curent din progresia aritmetică, cel care urmează a fi tipărit.

Algoritmul de rezolvare:

a. Variabile de intrare:

a - primul termen al progresiei;
r - rația progresiei;
n - numărul termenilor ce trebuie tipăriți.

b. Variabile de ieșire:

t - ia ca valori termenii progresiei aritmetice.

c. Variabile de lucru:

i - variabilă de ciclare.

d. Algoritmul propriu-zis:

```
Algoritmul PROGRESIE este:
Date a, r, n;
Fie t := a;
Pentru i = 1,n execută
    Tipărește "T(i)="; t
    Fie t := t+r
sf {pentru}
```

Programul Basic:

```
 1  REM  Programul 2: PROGRESIE
 2  REM  ************************
10  REM  Progresie aritmetica
11  REM  cu ratia r si primul
12  REM  termen a. Se tiparesc
13  REM  primii  n termeni.
15  REM  ************************
16  REM
20  INPUT "Ratia="; r
30  INPUT "Primul termen="; a
40  INPUT "Numarul termenilor="; n
50  LET  t = a
60  FOR i = 1 TO n
70      PRINT "T(" ; i ; ")=" ; t
80      LET  t = t + r
90  NEXT i
100 STOP
```

3.12. Programul 3: HORNER.

În continuare vom da un alt exemplu de program, care calculează cîtul şi restul din împărţirea polinomului:

P(1)*X↑n + ... + P(n)*X + P(n+1)

la (X-a), folosind schema lui Horner.

Algoritmul de rezolvare:

a. Variabile de intrare:
   - n - gradul polinomului;
   - (P(i), i=1,n+1) - vectorul coeficienţilor;
   - a - număr real.

b. Variabile de ieşire:
   - (Q(i),i=1,n) - vectorul coeficienţilor cîtului;
   - Q(n+1) - restul împarţirii (egal cu P(a)).

c. Variabile de lucru:
   - n1=n+1 - numărul coeficienţilor polinomului P;
   - i - variabilă de ciclare.

d. Algoritmul propriu-zis:

```
Algoritmul HORNER este:
Date n, (P(i),i=1,n+1) , a ;
Fie Q(1)=P(1);   n1 := n+1
Pentru i=2,n1 execută Q(i) := a * Q(i-1) + P(i)  sf {pentru}
Rezultate (Q(i), i=1,n), Q(n1)
```

Programul Basic:

```
1   REM   Programul 3: HORNER
2   REM **********************
10  REM Schema lui Horner
12  REM **********************
13  REM
20  INPUT "Gradul polinomului="; n
22  LET n1=n+1:
30  DIM  P(n1):    DIM  Q(n1)
40  FOR i = 1 TO n1
50      PRINT "P("; i ; ")=" ;
60      INPUT  P(i)
65      PRINT  P(i)
70  NEXT  i
80  INPUT "a=" ; a
```

```
90  LET Q(1) = P(1)
100 FOR i = 2 TO n1
110     LET Q(i) = a * Q(i-1) + P(i)
120 NEXT i
130 PRINT "Coeficientii citului:"
132 PRINT "din impartirea lui"
134 PRINT "P(X) la X-a sint:"
136 PRINT
140 FOR i = 1 TO n
150     PRINT "Q(" ; i ; ")=" ; Q(i)
160 NEXT i
165 PRINT
170 PRINT "Restul impartirii=" ; Q(n1)
180 STOP
```

Pentru execuţia programului se dă comanda **RUN**, la care calculatorul execută instrucţiunile în ordinea întîlnirii lor. Pe viitor la cele mai multe programe vom prezenta ceea ce apare pe ecran ca rezultat al execuţiei programului respectiv.

Menţionăm că informaţia subliniată se dă de către programator, restul informaţiei fiind afişată de calculator. Uneori se dau şi comentarii ale autorilor, ele fiind închise în paranteze drepte.

Rezultatul execuţiei:

Gradul polinomului=<u>3</u>
P(1)=<u>1</u>
P(2)=<u>1</u>
P(3)=<u>1</u>
P(4)=<u>11</u>
a=<u>-1</u>

[S-a terminat introducerea datelor. În urma execuţiei, pe ecran apare:]

P(1)=1
P(2)=1
P(3)=1
P(4)=11
a=-1

Coeficientii citului
din impartirea lui
P(X) la X-a sint

Q(1)=1
Q(2)=0
Q(3)=1

Restul impartirii=10

3.13. Subprograme Basic.

În programarea unor probleme mai complicate este posibil ca același subalgoritm să fie folosit de mai multe ori (în locuri diferite ale programului). Este de dorit ca acest subalgoritm să fie programat o singură dată, dar să fie folosit ori de cîte ori dorim acest lucru. Putem practica o asemenea programare folosind noțiunea de **subprogram**.

Un subprogram Basic are forma:

```
n1  prima instrucţiune  ┐
    .    .    .         │ A =  corpul subprogramului
    .    .    .         ┘
n2  RETURN
```

şi este apelat prin instrucţiunea:

```
n   GOSUB  n1
```

Instrucţiunea **GOSUB n1** provoacă un salt la instrucţiunea cu eticheta *n1*, analog cu cel provocat prin instrucţiunea **GOTO n1**. Diferenţa constă în faptul că **GOSUB** cere în plus memorarea numărului de linie a instrucţiunii care urmează după ea (imediat superior lui n). După execuţia corpului **A** al subprogramului, la prima întîlnire a instrucţiunii **RETURN**, se revine la instrucţiunea cu numărul de linie memorat, deci la prima instrucţiune care urmează după **GOSUB**. Menţionăm că în Basic subprogramele nu pot avea parametri formali.

Pe lîngă funcţiile implicit definite programatorul își poate defini propriile funcţii folosind instrucţiunea **DEF**. Ea are sintaxa:

n DEF FNf (*lista*) = *expresie*

unde *lista* este lista variabilelor simple de care depinde funcţia definită, iar expresie precizează calculele care trebuie făcute pentru a obţine valoarea funcţiei. Numele funcţiei definite este **FNf**, unde *f* este o literă aleasă de programator. O funcţie este apelată direct prin scrierea într-o expresie aritmetică a numelui funcţiei **FNf**, urmat între paranteze de expresii aritmetice care indică valorile fiecărei variabile.

3.14. Programul 4: POLINOM.

Ca exemplu de folosire a instrucţiunilor **GOSUB** şi **DEF** vom scrie un program care calculează valorile a două polinoame *P* şi *R* folosind schema lui Horner ca subalgoritm. Coeficienţii celor două polinoame sînt:

P(i) = F(2*i+1), R(i) = F(i*i+i+1), i=1,2,...,n,

unde *F* este o funcţie dată. Programul este următorul:

```
1   REM Programul 4: POLINOM
2   REM **********************
10  REM Valoarea a doua poli-
11  REM noame  P(X) si Q(X)
12  REM  pentru X dat
13  REM **********************
14  REM
20  DEF FNF (t) = EXP (-t/2) * t - SIN (t)
30  INPUT "Gradul polinomului=" ; n
40  INPUT "Valoarea lui x:" ; x
45  LET n1=n+1
50  DIM P(n1)
60  FOR i = 1 TO n1
70      LET P(i) = FNF (2*i+1)
80  NEXT i
90  GOSUB 200
100 PRINT "P(" ; x ; ")=" ; v
106 REM
110 REM Al doilea polinom este
111 REM   notat tot prin P
112 REM intrucit asa se cere in
```

```
113 REM    subprogram
120 FOR i = 1 TO n1
130       LET P(i) = FNF (i*i+i+1)
140 NEXT i
150 GOSUB 200
160 PRINT "R(" ; x ; ")=" ; v
170 STOP
198 REM
199 REM ********************************
200 REM Subprogram care calculeaza
201 REM        v = P(x)
202 REM bazat pe schema lui Horner
203 REM Se foloseste variabila de lucru j
204 REM ********************************
210 LET v = P (1)
220 FOR j = 2 TO n1
230       LET v = v * x + P (j)
240 NEXT j
250 RETURN
```

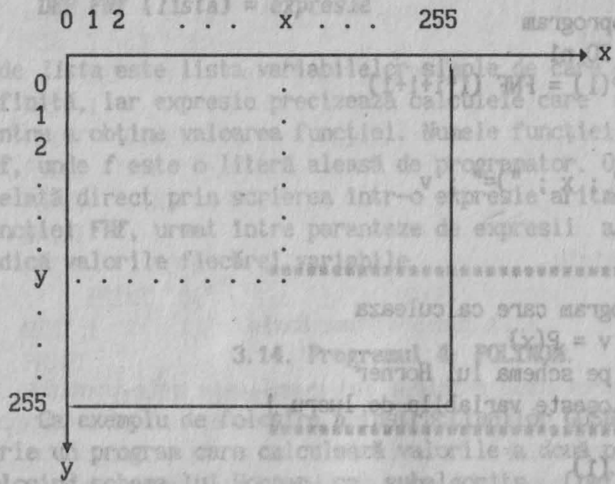
Rezultatul executiei:

```
Gradul polinomului=3
Valoarea lui x=5
P(5)=350
R(5)=110.25127
```

3.15. Instructiuni grafice in Basic.

Avînd în vedere diferențele mari ce există în utilizarea funcțiilor grafice ale diverselor calculatoare, vom indica în continuare instrucțiunile BASIC pentru cele două calculatoare mai des întîlnite la noi, sau cele compatibile cu ele.

3.15.1. **BASIC PRAE**. Pentru acest limbaj, ecranul calculatorului este împărțit în 256 linii (numerotate de la 0 la 255) și 256 coloane (numerotate tot de la 0 la 255). De aici deducem că ecranul este format din 256*256=65536 puncte. Precizarea acestor puncte se face cu ajutorul coordonatelor (x,y), unde x si y au valori din mulțimea {0,1,...,255}. Coordonatele diferitelor puncte de pe ecran se deduc din figura următoare:



Se observă că punctul de coordonate (0,0) se află în colţul din stînga şi sus al ecranului.

Pentru a putea trasa diferite grafice pe ecranul calculatorului, limbajul oferă următoarele şase instrucţiuni:

1. **PLOT** X,Y - pentru a desena pe ecran punctul de coordonate (X,Y);
2. **PLOTC** X,Y - pentru a şterge de pe ecran punctul de coordonate (X,Y);
3. **DRAW** X,Y - pentru a desena un segment de dreaptă ce uneşte ultimul punct desenat pe ecran (cu **PLOT** sau **DRAW**) cu punctul de coordonate (X,Y);
4. **DRAWC** X,Y - pentru a şterge un segment de dreapta ce uneşte ultimul punct şters de pe ecran cu punctul de coordonate (X,Y);
5. **CIRCLE** X,Y,R - pentru a desena un cerc cu centrul în punctul de coordonate (X,Y) şi raza egală cu R;
6. **CIRCLEC** X,Y,R - pentru a şterge cercul de rază R ce are centrul în punctul de coordonate (X,Y).

În aceste instrucţiuni, *X*,*Y* şi *R* pot fi:constante, variabile sau expresii aritmetice. După evaluare, ele se trunchiază la valoarea întreagă. Dacă în execuţia oricăreia din aceste instrucţiuni se ajunge la un punct ce nu aparţine ecranului, atunci se afişează mesajul **ILLEGAL FUNCTION** şi execuţia instrucţiunii se termină.

Ecranul calculatorului poate avea două culori: albă sau nea-

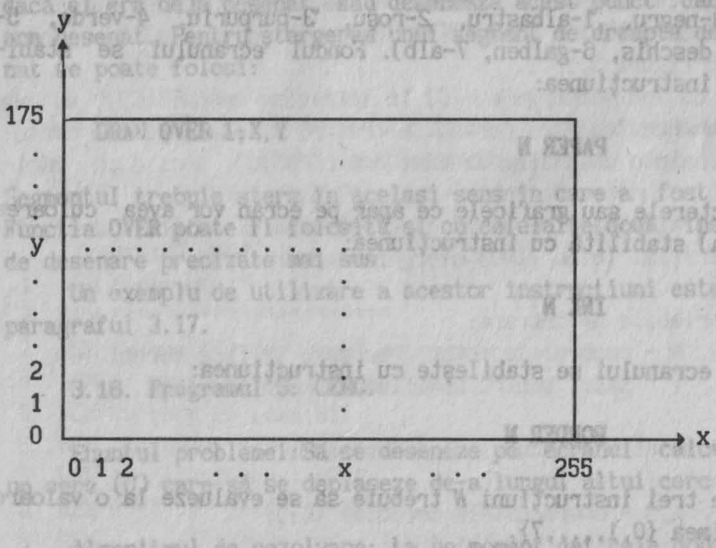
gră. Caracterele sau graficele ce apar pe ecran vor avea culoarea opusă fondului ecranului (dacă ecranul are culoarea albă, atunci caracterele sau graficele vor avea culoarea neagră și invers). Fondul ecranului se stabilește cu instrucțiunea **SWITCH V**. Dacă valoarea lui *V* este 1 atunci în continuare ecranul va avea culoarea neagră, iar dacă V=0 atunci ecranul va fi alb. Cu funcția

SWITCH se stabilește fondul ecranului pentru instrucțiunile ce se vor executa în continuare. Cu instrucțiunea **CLS** se poate șterge ecranul, după care fondul ecranului va avea culoare albă sau neagră și anume culoarea stabilită ultima dată.

Un exemplu de utilizare a acestor instrucțiuni se dă în paragraful 3.16.

3.15.2. **BASIC HC**. Acest limbaj consideră ecranul calculatorului format din 176 linii (numerotate de la 0 la 175) și 256 coloane (numerotate de la 0 la 255). Sub cele 176 linii grafice mai există 2 linii de caractere (din cele 24 linii posibile) unde se pot afișa mesaje, valori, sau de unde se pot cere date pentru diferite variabile.

Coordonatele punctelor de pe ecran sînt precizate în figura următoare:



Instrucţiunile grafice oferite de acest limbaj sînt:

1. PLOT X,Y - pentru a desena pe ecran punctul de coordonate (X,Y);
2. DRAW X,Y - dacă ultimul punct desenat are coordonatele (A,B), atunci instrucţiunea uneşte acest punct cu punctul de coordonate (A+X,B+Y) printr-un segment de dreaptă;
3. CIRCLE X,Y,R -pentru desenarea unui cerc cu centrul în punctul de coordonate (X,Y) şi rază R;
4. DRAW X,Y,M - dacă ultimul punct desenat pe ecran are coordonatele (A,B), atunci instrucţiunea uneşte acest punct cu punctul de coordonate (A+X,B+Y) printr-un arc de cerc ce are ca masură valoarea absolută a variabilei M. Raza cercului depinde de valorile X, Y, M, A şi B, iar sensul de unire a celor două puncte depinde de semnul lui M.

Valorile din aceste instrucţiuni pot fi: constante, variabile sau expresii aritmetice, care după evaluare se trunchiază la valoarea întreagă. Dacă în execuţia acestor instrucţiuni se ajunge la un punct ce nu aparţine ecranului, atunci se termină execuţia şi apare mesajul "Integer out of range".

Ecranul calculatorului poate avea 8 culori, numerotate de la 0 la 7 (0-negru, 1-albastru, 2-roşu, 3-purpuriu, 4-verde, 5-albastru deschis, 6-galben, 7-alb). Fondul ecranului se stabileşte cu instrucţiunea:

PAPER N

iar caracterele sau graficele ce apar pe ecran vor avea culoarea (cerneala) stabilită cu instrucţiunea:

INK N

Marginea ecranului se stabileşte cu instrucţiunea:

BORDER N

In aceste trei instrucţiuni *N* trebuie să se evalueze la o valoare din mulţimea {0,1,...,7}.

Într-o zonă ecran pe care în mod obişnuit se afişează un ca-

racter (o zonă de 8*8 puncte în mod grafic) putem avea o singură culoare pentru fondul ecranului și o singură culoare pentru caracterele sau graficele ce apar. Instrucțiunile PAPER și INK stabilesc culorile pentru instrucțiunile ce se vor executa în continuare.

Ecranul calculatorului se poate șterge cu instrucțiunea CLS.

Pentru controlul tipăririi de caractere sau desenării de grafice, se pot folosi și următoarele funcții:

FLASH V - pentru a realiza o pîlpîire a ceea ce se tipărește (o inversare periodică a culorii cernelii cu a ecranului). Cu V=1 se activează pîlpîirea, iar cu V=0 se dezactivează;

INVERSE 1 - pentru a inversa culoarea ecranului cu cea a cernelii;

OVER 1 - pentru a se face o supratipărire: noile informații apar peste cele existente deja pe ecran.

Funcția OVER se folosește și pentru a face ștergerea diferitelor grafice de pe ecran, dacă se utilizează în cadrul celor patru funcții de desenare date mai sus. Astfel, funcția:

```
PLOT OVER 1;X,Y
```

realizează o ștergere de pe ecran a punctului de coordonate (X,Y) dacă el era deja desenat, sau desenează acest punct dacă el nu era desenat. Pentru ștergerea unui segment de dreaptă deja desenat se poate folosi:

```
DRAW OVER 1;X,Y
```

Segmentul trebuie șters în același sens în care a fost desenat. Funcția **OVER** poate fi folosită și cu celelalte două instrucțiuni de desenare precizate mai sus.

Un exemplu de utilizare a acestor instrucțiuni este dat în paragraful 3.17.

3.16. Programul 5: CERC.

Enunțul problemei: Să se deseneze pe ecranul calculatorului un cerc (*C*) care să se deplaseze de-a lungul altui cerc (*C1*).

Algoritmul de rezolvare: La un moment dat cele două cercuri se desenează pe ecran în modul indicat pe figura 3.16.1

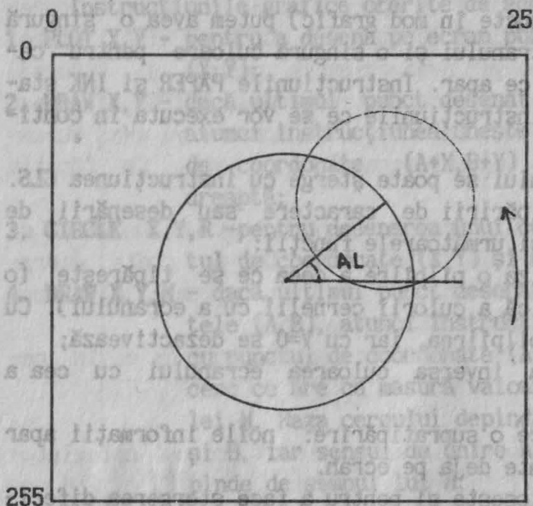


Fig.3.16.1.

Dacă cercul (*C1*) are centrul de coordonate (A,B) și raza de lungime *R1*, atunci coordonatele centrului cercului (*C*), determinat de unghiul *AL* vor fi egale cu:

X=A+R1*COS(AL),

Y=B-R1*SIN(AL).

În calculul lui *Y* s-a luat semnul "-" deoarece valoarea lui Y se micșorează cu valoarea lui *SIN(AL)* cînd ne deplasăm în sensul indicat în figura 3.16.1.

După ce vom parcurge cu (*C*) în întregime cercul (*C1*), se va face schimbarea fondului ecranului. Pentru a reda mișcarea cercului (*C*), înainte de desenarea unui nou cerc (*C*), deci după mărirea lui *AL* cu un pas *V*, vechiul cerc (*C*) se va șterge de pe ecran. Această operație se efectuează dacă pe ecran el există, deci la momentul inițial (cînd *AL*=0) ștergerea nu are sens.

a. Variabile de intrare:

N - numărul de rotații pe care le face cercul (C);

V - pasul pentru modificarea unghiului AL;

b. Rezultate: desenul indicat în enunț.

c. Variabile de lucru:

A,B - coordonatele cercului (C1);

R1 - raza cercului (C1);

X,Y - coordonatele cercului (C);

R - raza cercului (C);
X0,Y0 - coordonatele ultimului cerc desenat;
AL - unghiul de rotaţie;
K - ia valoarea 0 sau 1, pentru a indica ce culoare va avea ecranul;
I - variabilă de ciclare.

d. Algoritmul propriu - zis:

1. A:=125; B:=130; [mijlocul aproximativ al ecranului va fi centrul cercului (C1)]
   R1:=75; R:=30;
2. Citeşte valorile lui N şi V;
3. K:=1; [pentru prima rotaţie culoarea ecranului va fi neagră]
4. Pentru I = 1,N execută paşii 4.1-4.3:
   4.1. Culoarea ecranului se stabileşte după valoarea lui K;
   Şterge ecranul;
   4.2. Pentru AL = 0 , 6.283 [egal cu 2*PI] , V execută paşii 4.2.1-4.2.4:
      4.2.1. X:=A+R1*COS(AL);
      Y:=B-R1*SIN(AL);
      4.2.2. Dacă AL <> 0 atunci şterge cercul cu centrul în (X0,Y0) şi raza R;
      4.2.3. Desenează cercul cu centrul în (X,Y) şi raza R;
      4.2.4. X0:=X; Y0:=Y;
   sf {pentru}
   4.3. K:=1-K; [următoarea rotaţie se va face cu ecranul de culoare complementară]
5. Sfîrşitul algoritmului.

Programul BASIC:

```
1 REM Programul 5: CERC
2 REM ************************
3 REM Cu centrul pe cercul de raza R
4 REM si centru (A,B) se deplaseaza
5 REM un cerc de raza R1.
6..REM Programul a fost executat
7 REM la calculatorul PRAE
8 REM************************
9 REM
10 A=125: B=130: R1=75: R=30
```

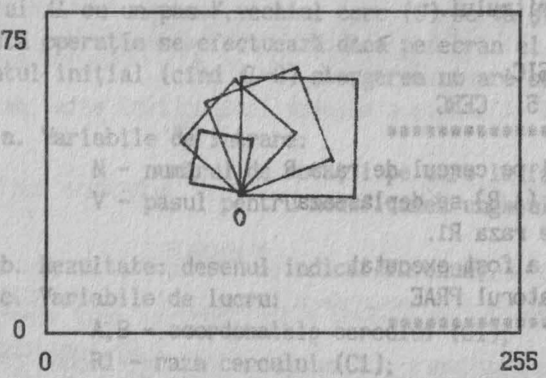
```
20 INPUT "N=";N: INPUT "V=";V
30 K=1
40 FOR I=1 TO N
50   SWITCH K: CLS
60   FOR AL=0 TO 6.283 STEP V
70     X=A+R1*COS(AL):   Y=B-R1*SIN(AL)
80     IF AL<>0 THEN CIRCLEC X0,Y0,R
90     CIRCLE X,Y,R
100    X0=X: Y0=Y
110  NEXT AL
120  K=1-K
130 NEXT I
140 STOP
```

Mişcarea este sugerată dacă introducem pentru V o valoare apropiată de 0.2.

3.17. Problema 6: PATRAT.

Enunţul problemei: Să se deseneze pe ecranul calculatorului un şir de pătrate: P(1),P(2),...,P(n). Un pătrat P(i+1) se obţine din pătratul anterior P(i) prin modificarea laturii cu o valoare *PL* şi rotaţia cu un unghi *PU* în jurul originii axelor de coordonate. Acest ciclu de desenare să se repete de *M* ori, cu culori diferite ale ecranului şi cernelii.

Algoritmul de rezolvare: La un moment dat desenul ar putea arăta ca în figura următoare, dacă pătratul P(1) are un vîrf în originea axelor de coordonate (punctul O).



Presupunem că pătratul P(1) are vîrfurile în punctele de coordonate (A(i),B(i)), i=1,4, deci latura pătratului este egală cu L=SQR((A(2)-A(1))↑2+(B(2)-B(1))↑2). Pătratul P(i) se obține atunci din pătratul P(1) prin rotirea lui P(1) cu un unghi de măsura AL=(i-1)*PU și a cărui latură va fi L1=L+(i-1)*PL. Coordonatele (X(j),Y(j)) ale celor patru puncte ce formează pătratul P(i) se obțin după relațiile:

X(j)=[L1/L*(A(j)*COS(AL)-B(j)*SIN(AL))+0.5],
Y(j)=[L1/L*(A(j)*SIN(AL)+B(j)*COS(AL))+0.5],

deoarece facem și o rotunjire a coordonatelor obținute.

Pentru desenarea unui pătrat oarecare, ce are vîrfurile în punctele de coordonate (X(j),Y(j)),j=1,4, vom considera și punctul de coordonate (X(5),Y(5))=(X(1),Y(1)), și desenăm patru segmente de dreaptă, delimitate de punctele:

(X(j-1),Y(j-1)) și (X(j),Y(j)), j=2,5.

Pentru a centra cît mai bine desenele pe ecran, vom plasa originea axelor de coordonate în punctul din centrul ecranului, deci în punctul (U,V)=(125,87).

a. Variabile de intrare:
- A(i),B(i), i=1,4 - coordonatele vîrfurilor primului pătrat;
- PL - pasul de modificare a lungimii laturii pentru două pătrate consecutive;
- PU - pasul de modificare a unghiului de rotație pentru două pătrate consecutive;
- N - numărul de pătrate ce se desenează;
- M - numărul de cicluri de desenare;

b. Rezultate: desenul indicat în enunț.

c. Variabile de lucru:
- A(5),B(5) - pentru păstrarea coordonatelor primului punct;
- X(j),Y(j),j=1,5 - coordonatele vîrfurilor pătratului ce se desenează; aici (X(5),Y(5)) = (X(1),Y(1));
- L - lungimea laturii pătratului inițial;

L1 - lungimea laturii pătratului ce se desenează;
E - culoarea ecranului;
C - culoarea cernelii;
(U,V) - poziţia pe ecran a originii axelor de coordonate
I,J,K - variabile de lucru pentru controlul ciclurilor.

d. Algoritmul propriu-zis:

```
1. Date (A(i),B(i), i=1,4), PL,PU,N,M;
2. L:=SQR((A(2)-A(1))↑2+(B(2)-B(1))↑2);
   A(5):=A(1); B(5):=B(1);
   E:=0; C:=4;
   U:=125; V:=87;
3. Pentru i = 1 , M execută paşii 3.1-3.3:
   3.1. Stabileşte fondul ecranului la culoarea E şi cerneala la
        culoarea C;    Şterge ecranul;
   3.2. Pentru k = 1, N execută paşii 3.2.1-3.2.5:
        3.2.1. L1:=L+(K-1)*PL; AL:=(K-1)*PU;
        3.2.2. Pentru j = 1 , 4 execută
                  X(j):=[L1/L*(A(j)*COS(AL)-B(j)*SIN(AL))+0.5]+U
                  Y(j):=[L1/L*(A(j)*SIN(AL)+B(j)*SIN(AL))+0.5]+V
               sf {pentru}
        3.2.3. X(5):=X(1); Y(5):=Y(1);
        3.2.4. Desenează punctul de coordonate (X(1),Y(1));
        3.2.5. Pentru j = 2 , 5 execută
                    Desenează segmentul de dreapta ce uneşte
                    punctul anterior desenat cu (X(j),Y(j))
               sf {pentru}
        sf {pentru};
   3.3. E:=E+1; dacă E>7 atunci E:=0;
        C:=C+1; dacă C>7 atunci C:=0;
   sf {pentru}
```

Programul BASIC:

```
1  REM  Problema 6: PATRAT
2  REM ********************
3  REM deseneaza n patrate
4  REM ********************
5  REM
10 DIM A(5): DIM B(5): DIM X(5): DIM Y(5)
```

```
20 FOR I=1 TO 4
30 PRINT "Coord.punctului nr.";I
40 INPUT "A=";A(I),"B=";B(I)
50 NEXT I
60 INPUT "Pas modif.latura ";PL
70 INPUT "Pas unghi(in grade) ";PU
80 LET PU=PI*PU/180
81 REM PU va fi in radiani
90 INPUT "Nr.de patrate ";N
100 INPUT "Nr.de cicluri ";M
110 LET L=SQR((A(2)-A(1))*(A(2)-A(1))+(B(2)-B(1))*(B(2)-B(1)))
120 LET A(5)=A(1): LET B(5)=B(1)
130 LET E=0: LET C=4
140 LET U=125: LET V=87
150 FOR I=1 TO M
160   PAPER E: INK C: CLS
170   FOR K=1 TO N
180     LET L1=L+(K-1)*PL:     LET AL=(K-1)*PU
190     FOR J=1 TO 4
200       LET X(J)=INT(L1/L*(A(J)*COS(AL)-B(J)*SIN(AL))+0.5)+U
210       LET Y(J)=INT(L1/L*(A(J)*SIN(AL)+B(J)*COS(AL))+0.5)+V
220     NEXT J
230     LET X(5)=X(1):         LET Y(5)=Y(1)
240     PLOT X(1),Y(1)
250     FOR J=2 TO 5
260       DRAW X(J)-X(J-1),Y(J)-Y(J-1)
270     NEXT J
280   NEXT K
290   LET E=E+1: IF E>7 THEN LET E=0
300   LET C=C+1: IF C>7 THEN LET C=0
310 NEXT I
320 STOP
```

Se obţin desene interesante pentru următoarele date:

(A(1),B(1))	(A(2),B(2))	(A(3),B(3))	(A(4),B(4))	PL	PU	N	M
(0,0)	(60,0)	(60,60)	(0,60)	-2.5	7.5	20	10
(0,0)	(60,0)	(60,60)	(0,60)	-10	0	13	10
(50,-50)	(50,50)	(-50,50)	(-50,-50)	-4	7.5	20	10
(50,-50)	(50,50)	(-50,50)	(-50,-50)	0	7.5	12	10

3.18. Operatii asupra sirurilor de caractere.

In limbajul Basic există variabile simple sau indexate de tip şir de caractere. Ele se deosebesc de variabilele numerice prin faptul că identificatorul lor se termină întotdeauna cu caracterul "$" (dolar).

Tablourile de tip şir de caractere se declară tot prin instrucţiunea DIM. Este necesară rezervarea unei zone de memorie a cărei mărime trebuie cunoscută si anume o locaţie pentru fiecare variabilă indexată.

Astfel, dacă dorim să păstrăm în calculator numele elevilor unei clase cu cel mult 40 de elevi şi fiecare nume are cel mult 20 de caractere, vom avea declaraţia:

DIM N$(40,20)

Un exemplu de program în care se foloseşte această declaraţie şi se fac multe alte operaţii cu şiruri de caractere se dă în 4.13.

3.19. Alte instructiuni Basic.

Există situaţii cînd anumite date sînt aceleaşi la mai multe execuţii ale unui program şi ar fi neplăcut să le scriem de fiecare dată în vederea introducerii lor în calculator. În aceste situaţii există posibilitatea scrierii lor deodată cu programul, folosind instrucţiunea **DATA**.

Instrucţiunea **DATA** permite crearea unei zone de date dispuse în memoria calculatorului, date care vor fi citite prin instrucţiunea **READ**. Formatul instrucţiunii este:

DATA *lista*

unde "*lista*" este o listă de constante separate prin virgulă.

Instrucţiunea **READ** are sintaxa:

READ *lista*

unde "*lista*" conţine variabile simple sau indexate, separate prin virgulă. Ea cere atribuirea unei valori fiecărei variabile din listă. Această valoare este constanta care urmează în blocul de date. Rezultă că între constantele

c(1), c(2), ... , c(n)

aflate în această ordine în instrucţiunile DATA din program şi variabilele

v(1), v(2), ... , v(m)

întîlnite (în această ordine) în instrucţiunile **READ** se stabileşte o corespondenţă. Atribuirea:

v(i) := c(i)

va avea loc numai dacă variabila *v(i)* şi constanta *c(i)* sînt de acelaşi tip. Programul se va termina cu eroare în cazul în care *n*<*m*, eroare ce constă în insuficienţa datelor necesare instrucţiunilor **READ** din program.

Un exemplu de folosire a acestor instrucţiuni se dă în 4.19.

Instrucţiunea **RESTORE** permite refolosirea unor date introduse în program prin instrucţiunile **DATA**. Ea are sintaxa:

RESTORE [**numar** de linie]

şi cere ca în timpul execuţiei programului, la prima citire cu instrucţiunea **READ** care urmează, datele folosite să fie cele din instrucţiunea **DATA** cu numărul de linie specificat după cuvîntul **RESTORE**. În cazul cînd numărul de linie lipseşte, se va considera că este vorba de prima instrucţiune **DATA** din program.

4. ALTE PROGRAME.

Dăm în continuare mai multe exemple de programe Basic executate la calculatorul HC-85, pentru probleme întîlnite în manualele şcolare. Fiecare exemplu conţine enunţul problemei, urmat de un algoritm de rezolvare a problemei, apoi de programul Basic şi de un exemplu de execuţie a programului.

În aceste exemple informaţia subliniată se dă de către utilizator de la tastatura calculatorului. Restul informaţiilor date la exemplul de utilizare sînt tipărite de calculator.

4.1. Programul 7 : ECUATIE2.

Enunţul problemei : Să se rezolve ecuaţia de gradul doi:

a x↑2 + b x + c = 0

Algoritmul de rezolvare:

a. Variabile de intrare : a, b, c - coeficienţii ecuaţiei.
b. Variabile de ieşire :
   x1, x2 - rădăcinile ecuaţiei dacă sînt reale, respectiv partea reală şi imaginară a rădăcinilor complexe.
c. Variabile de lucru : D - discriminantul ecuaţiei;
d. Algoritmul propriu-zis :

```
Date a,b,c;
Dacă a=0 atunci tipăreşte mesaj "Ec. nu este de gradul doi"
                Stop
sf {dacă}
```

```
Dacă a ≠ 0 atunci
      Fie D:=b↑2-4*a*c;
      Dacă D<0
          atunci tipăreşte "D<0;răd.complexe conjugate"
                  Fie X1:=-b/(2*a)
                  Fie X2:=SQR(-D)/(2*a)
          altfel dacă D=0 atunci mesaj "D=0;răd.reale confundate"
                              Fie X1:=X2:=-b/2*a .
                         altfel Fie X1:=(-b+ SQR(D))/(2*a)
                              Fie X2:=(-b- SQR(D))/(2*a)
                           tipăreşte "D>0;răd.reale distincte"
                  sf {dacă}
      sf {dacă}
      Rezultate X1, X2
sf {dacă}
```

Programul BASIC:

```
1   REM  Programul 7:ECUATIE2
2   REM  **********************
10  REM  REZOLVAREA  ECUATIEI
12  REM  DE GRAD II
13  REM  a x↑2 + b x + c = 0
14  REM  **********************
20  PRINT "COEFICIENTII ECUATIEI:"
30  INPUT "a=";a
35  INPUT "b=";b
40  INPUT "c=";c
50  CLS: PRINT
55  PRINT "COEFICIENTII ECUATIEI SINT"
60  PRINT "a=";a
70  PRINT "b=";b
80  PRINT "c=";c
90  IF a=0 THEN GO TO 300
100 LET D=b*b-4*a*c
110 PRINT "VALOAREA DISCRIMINANTULUI ESTE
112 PRINT "     D=";D
120 IF D<0 THEN GO TO 240
130 IF D=0 THEN GO TO 200
140 PRINT:PRINT "D>0 , RADACINI REALE DISTINCTE"
150 LET X1=(-b+SQR D)/(2*a)
160 LET X2=(-b-SQR D)/(2*a)
```

```
170 PRINT:PRINT "X1=";X1
180 PRINT "X2=";X2
190 GO TO 310
200 PRINT:PRINT "D=0, RADACINI REALE CONFUNDATE"
210 LET X1=-b/(2*a): LET X2=X1
230 GO TO 170
240 PRINT:PRINT "D<0, RADACINI COMPLEXE CONJUGATE"
250 LET X1=-b/(2*a)
260 LET X2=SQR(-D)/(2*a)
270 PRINT:PRINT "Partea reala="; X1
280 PRINT "Partea imaginara="; X2
290 GO TO 310
300 PRINT:PRINT "ECUATIA NU ESTE DE GRADUL DOI"
308 REM
310 PRINT:PRINT "DATI ALTE VALORI (1=DA,0=NU)?":
    INPUT N
320 IF N=1 THEN CLS: GO TO 20
330 STOP
```

Exemplu de utilizare:

```
COEFICIENTII ECUATIEI:
a=?23
b=?3
c=?-13
                              [se şterge ecranul şi urmează:]

COEFICIENTII ECUATIEI SINT
a=23
b=3
c=-13

VALOAREA DISCRIMINANTULUI ESTE
          D=1205

D>0,RADACINI REALE DISTINCTE

X1=0.68941543
X2=-0.81985022

DATI ALTE VALORI (1=DA,0=NU)?0
```

4.2. Programul 8 : COMBI.

Enunţul problemei: Fie *n* şi *k* două numere întregi pozitive date. Să se calculeze aranjamente şi combinări de *n* elemene luate cîte *k* şi permutări de *n* elemente.

Algoritmul de rezolvare:

a. Variabile de intrare:
    n, k - numere naturale

b. Variabile de ieşire:
    A - pentru A(n,k);
    C - pentru C(n,k);
    P - pentru P(n).

c. Variabile de lucru :
    A1 - pentru valoarea k!;
    A2 - pentru valoarea n!;
    A3 - pentru valoarea (n-k)!;

d. Algoritmul propriu-zis:

```
Date n ,k;
Dacă n>33 atunci tipăreşte "P(n) prea mare"; Stop  sf {dacă}
Dacă n<0 sau n ≠ [n] sau k<0 sau k ≠ [k]
    atunci tipăreşte "valoare imposibilă pt.n(sau k)";
          Stop
sf {dacă}
Dacă n=0 atunci tipăreşte "A,C,P nedefinite";
      altfel fie A2:=FACT(n)
           dacă k=0 atunci tipăreşte "A(n,k)=1
                                     C(n,k)=1"
                   altfel dacă k>n
                           atunci tipăreşte "Functii
                                             nedefinite"
                           altfel fie A1:=FACT(k)
                                      A3:=FACT(n-k)
                                      A := A2/A1
                                      C := A/A3
                        sf {dacă}
           sf {dacă}
           Fie P := A2
           Rezultate  A, C, P
sf {dacă}
```

```
Funcţia FACT(n) este:
    Fie FACT:= 1;
    Dacă n>0 atunci
        Pentru i=1,n execută  FACT:=FACT * i  sf {pentru}
    sf {dacă}
Revenire;
```

Programul BASIC:

```
1   REM  Programul 8: COMBI1
2   REM ************************
10  REM Analiza combinatorie
11  REM ************************
20  PRINT:PRINT "DATELE DE INTRARE"
40  INPUT "n=";n:   INPUT "k=";k
50  PRINT: PRINT "n=";n
60  PRINT "k=";k
70  IF n>33 THEN PRINT: PRINT "P(";n;") PREA MARE":
                GO TO 290
80  IF n<0 OR k<0 OR n<>INT n OR k<> INT k THEN GO TO 280
90  IF n=0 THEN PRINT:PRINT:
            PRINT "FUNCTIILE A,C si P NEDEFINITE": GO TO 290
99  REM             n!
100 LET R=n
110 GOSUB 380
120 LET A2=F
130 IF k=0 THEN GO TO 320
140 IF k>n THEN PRINT:PRINT:PRINT "k>n - FUNCTII NEDEFINITE":
                GO TO 290
149 REM             k!
150 LET R=k
160 GOSUB 380
170 LET A1=F
179 REM     (n-k)!
180 LET R=n-k
190 GOSUB 380
200 LET A3=F
210 LET A=A2/A3
220 LET C=A2/(A1*A3)
230 PRINT:PRINT:PRINT "A(";n;",";k;")=";A
```

```
240 PRINT:PRINT "C(";n;",";k;")=";C
250 LET P=A2
260 PRINT:PRINT "P(";n")=";P
270 GOTO 290
280 PRINT:PRINT "VALOARE IMPOSIBILA PENTRU n (SAU k)"
288 REM
290 PRINT:PRINT "DATI ALTE VALORI (1=DA,0=NU)?":
    INPUT N
300 IF N=1 THEN CLS:GO TO 20
310 STOP
320 PRINT:PRINT:PRINT "A(";n;",";k;")=";1
330 PRINT:PRINT "C(";n;",";k;")=";1
370 GO TO 250
377 REM            FACT(R)
378 REM          ************
380 LET F=1
390 FOR L=1 TO R
400 LET F=F*L
410 NEXT L
420 RETURN
```

Exemplu de utilizare:

```
DATELE DE INTRARE
n=?10
k=?5

A(10,5)=30240
C(10,5)=252
P(10)=3628800

DATI ALTE VALORI (1=DA,0=NU)?0
```

Pentru n>33 programul **COMBI1** nu mai poate fi utilizat întrucît valoarea *n!* nu poate fi reprezentată în calculator. Chiar şi pentru *n* mai mic decît *33*, valoarea *n!* se reţine în calculator aproximativ. Cu toate acestea, valorile *C(n,k)* şi *A(n,k)* ar putea fi calculate folosind o relaţie iterativă. Dăm în continuare un program care calculează *C(n,k)* bazat pe formula:

C(n,j) = C(n,j-1)*(n-j+1)/j,   j=1,2,...,k.

Vom folosi următorul algoritm:

Algoritmul COMBI2 este:
Date n,k;
Dacă n $\notin$ N, sau k $\notin$ N, atunci tipăreşte "Date greşite"; Stop
sf {dacă}
Dacă n<k atunci tipăreşte "Funcţii nedefinite"
    altfel fie C:=1;
        Pentru j=1,k execută C:=C*(n-j+1)/j sf {pentru}
        Tipăreşte "C(n,k)="; C
sf {dacă}

Programul Basic:

```
1   REM  Programul 8B: COMBI2
2   REM  **********************
10  REM Programul calculeaza
11  REM C(n,k) pentru n,k dati
12  REM **********************
20  INPUT "n=";n
30  INPUT "k=";k
40  IF n<0 OR k<0 THEN GOTO 300
50  IF n<> INT(n) THEN GOTO 300
60  IF k<> INT(k) THEN GOTO 300
70  IF n=0 OR k>n THEN GOTO 400
80  LET  C=1
90  FOR j=1 TO k
100     LET C=C*(n-j+1)/j
110 NEXT j
120 PRINT "C("; n; ","; k; ")="; C
130 STOP
300 PRINT "Valori incorecte pentru n sau k"
310 PRINT "n="; n
320 PRINT "k="; k
330 STOP
399 REM
400 PRINT "Functie nedefinita"
410 GOTO 310
```

Exemplu de utilizare:

```
n=?10
k=?5

C(10,5)=252
```

În sfîrşit, o a treia posibilitate de a calcula *C(n,k)* se bazează pe relaţia:

C(n,k) = C(n-1,k) + C(n-1,k-1),

cu ajutorul căreia se poate construi triunghiul lui Pascal:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
 . . . . .
```

Algoritmul de rezolvare:

a. Variabile de intrare:

n şi k - numere naturale cu n>=k.

b. Variabile de ieşire:

N(k+1) - ce are valoarea C(n,k) la sfîrşitul execuţiei.

c. Variabile de lucru:

V - Linia veche (existentă) din triunghiul lui Pascal;

N - Linia nouă care se calculează din V;

i,j - indici de ciclare.

d. Algoritmul propriu-zis:

```
Date n,k;
Pentru j=2,n+1 execută V(j):=0 sf {pentru}
Fie V(1):=1, Fie N(1):=1
Pentru i=2,n-1 execută
    Pentru j=2,i+1 execută N(j):=V(j)+V(j-1) sf {pentru}
    Pentru j=2,i+1 execută V(j):=N(j) sf {pentru}
sf {pentru}
Pentru j=2,k+1 execută N(j):=V(j)+V(j-1) sf {pentru}
Rezultate "C(n,k)="; N(k+1)
```

Programul Basic:

```
1    REM Programul 8C: COMBI3
2    REM ********************
10   REM Triunghiul lui Pascal
12   REM Se tipareste numai
14   REM     C(n,k)
16   REM pentru n,k date
18   REM ********************
30   INPUT "n="; n
```

```
40  INPUT "k="; k
44  IF n<2 THEN GOTO 200
45  IF k<0 OR k>n THEN GOTO 200
50  DIM V(n+1):  DIM N(n+1)
55  CLS
60  FOR j=2 TO n+1
62      LET V(j)=0
64  NEXT j
66  LET V(1)=1: LET N(1)=1
80  FOR i = 2 TO n-1
85      PRINT AT 2,14;"i=";i
90      FOR j=2 TO i+1
95          LET N(j)=V(j)+V(j-1)
100     NEXT j
110     FOR j=2 TO i+1
115         LET V(j)=N(j)
120     NEXT j
130 NEXT i
140 FOR j=2 TO k+1
145     LET N(j)=V(j)+V(j-1)
150 NEXT j
159 PRINT
160 PRINT "C("; n; " , "; k; " )="; N(k+1)
170 STOP
200 PRINT "Date gresite"
210 PRINT "n=";n; "  k=";k
220 GOTO 30
```

Exemplu de utilizare:

```
n=?10
k=?5
            i=2
            i=3
                [şi toate valorile pînă la]
            i=9
C( 10 , 5 )= 252
```

4.3. Programul 9: MATRICE.

Enunţul problemei: Se dă un număr *Kod* ∈ {1, 2} şi două matrice *A* şi *B*. Să se calculeze matricea *C* = *A* o *B* unde "o" este operaţia de adunare dacă *Kod*=1 şi înmulţire dacă *Kod*=2.

Algoritmul de rezolvare:

a. Variabile de intrare:

Kod - întreg egal cu 1 sau 2 pentru a indica operaţia ce se efectuează;
M1 - numărul liniilor matricei A;
N1 - numărul coloanelor matricei A;
M2 - numărul liniilor matricei B;
N2 - numărul coloanelor matricei B;
A, B - matrice de dimensiunile indicate.

b. Variabile de ieşire:

C - o matrice cu M1 linii şi N2 coloane.

c. Variabile de lucru:

i,j,k - variabile de ciclare.

d. Algoritmul propriu-zis:

```
Date Kod, M1,N1,M2,N2, A, B;
Dacă Kod=1 atunci
     Dacă M1≠M2 sau N1≠N2
          atunci tipăreşte "Date gresite"; Stop
          altfel
             Pentru i=1,M1 execută
                   Pentru j=1,N1 execută c(i,j):=a(i,j)+b(i,j)
                   sf {pentru}
             sf {pentru}
     sf {dacă}
sf {dacă}
Dacă Kod=2 atunci
     Dacă N1≠M2 atunci tipăreşte "Date gresite"; stop
             altfel Pentru i=1,M1 execută
                        Pentru j=1,N2 execută Fie c(i,j):=0;
                            Pentru k=1,N1 execută
                                c(i,j):=c(i,j)+a(i,k)*b(k,j)
                            sf {pentru}
                        sf {pentru}
                    sf {pentru}
     sf {dacă}
sf {dacă}
Rezultate ((c(i,j),j=1,N2), i=1,M1)
```

Programul Basic este:

```
1   REM   Programul 9:   MATRICE
2   REM  ****************************
10  REM  Se dau matricele A si B
11  REM  Pt. Kod=1 se calculeaza
12  REM       C  =  A + B
13  REM  Pt. Kod=2 se calculeaza
14  REM       C  =  A * B
15  REM  ****************************
20  INPUT "Kod="; Kod
25  INPUT "Nr.linii A=";M1
30  INPUT "Nr.coloane A=";N1
35  INPUT "Nr.linii B=";M2
40  INPUT "Nr.coloane B=";N2
42  IF Kod=1 AND (M1<>M2 OR N1<>N2) THEN GOTO 300
43  IF Kod=2 AND (N1<>M2) THEN GOTO 300
45  DIM A(M1,N1)
48  DIM B(M2,N2): DIM C(M1,N2)
50  FOR i=1 TO M1
55      FOR j=1 TO N1
60          PRINT "A(";i;",";j;")=";: INPUT A(i,j)
63          PRINT A(i,j)
65      NEXT j
70  NEXT i
75  PRINT
80  FOR i=1 TO M2
85      FOR j=1 TO N2
90          PRINT "B(";i;",";j;")=";: INPUT B(i,j)
93          PRINT B(i,j)
95      NEXT j
100 NEXT i
110 IF Kod=2 THEN GOTO 200
115 IF Kod<>1 THEN GOTO 500
120 REM Adunare
130 FOR i = 1 TO M1
140   FOR j = 1 TO N1
150     LET C(i,j) = A(i,j) + B(i,j)
160   NEXT j
170 NEXT i
180 GOTO 400
200 REM            Produs
```

```
210 FOR i= 1 TO M1
220   FOR j=1 TO N2
230     LET C(i,j)=0
240     FOR k=1 TO N1
250          LET C(i,j)=C(i,j)+A(i,k)*B(k,j)
260     NEXT k
270   NEXT j
280 NEXT i
290 GOTO 400
299 REM              date eronate
300 PRINT: PRINT "Date gresite"
310 PRINT "Matricea A are"
312 PRINT M1;" linii"
314 PRINT N1;" coloane"
320 PRINT:PRINT "Matricea B are"
322 PRINT M2;" linii"
324 PRINT N2;"coloane"
330 STOP
400 REM           Tiparirea rezultatelor
405 PRINT:PRINT "Matricea C este:"
410 FOR i=1 TO M1
420   FOR j=1 TO N2
430      PRINT "C(";i; ",";j; ")="; C(i,j)
440   NEXT j
450 NEXT i
460 STOP
500 PRINT "Valoarea lui Kod e gresita"
510 PRINT "  Kod = "; Kod
520 STOP
```

Exemplu de utilizare.

```
Kod=?1
Nr.linii A=?2
Nr.coloane A=?2
Nr.linii B=?2
Nr.coloane B=?2
A(1,1)=?11
A(1,2)=?12
A(2,1)=?21
A(2,2)=?22
```

B(1,1)=?<u>100</u>
B(1,2)=?<u>100</u>
B(2,1)=?<u>200</u>
B(2,2)=?<u>200</u>

Matricea C este:
C(1,1)=111
C(1,2)=112
C(2,1)=221
C(2,2)=222

4.4. Problema 10. NRE.

Enunţul problemei: Se tipăresc primii *n* termeni ai şirului

$$a(n) = \frac{1}{0!} + \frac{1}{1!} + \ldots + \frac{1}{n!},$$

convergent la numărul *e*.

Algoritmul de rezolvare: Pentru a calcula termenii acestui şir se foloseşte relaţia de recurenţă *a(i) = a(i-1) + t(i)*, unde

t(i) = 1/i! = t(i-1)/i.

Întrucît nu se cere să reţinem în memoria calculatorului termenii şirului, vom nota prin a ultimul element *a(i)* calculat şi prin *t* ultimul *t(i)* calculat.

a. Variabile de intrare:
n - numărul termenilor doriţi;

b. Variabile de ieşire:
a - va lua consecutiv valorile termenilor:
a(0), a(1), a(2), ... , a(n);

c. Variabile de lucru:
t - valoarea lui t(i);
i - contor ce numără termenii calculaţi;

d. Algoritmul propriu-zis:

```
Se citeşte valoarea lui n;
Fie a:= 1; t:= 1;
Pentru i = 1,n execută
    Fie t := t/i;
    Fie a:= a+t;
    Tipăreşte "a("; i ; ")= "; a
sf {pentru}
```

Programul Basic:

```
1   REM Programul 10: NRE
2   REM ********************
10  REM Se tiparesc primii n
11  REM termeni ai unui sir
12  REM convergent la
13  REM   numarul   e
14  REM ********************
20  INPUT "n="; n
30  LET a=1
40  LET t=1
50  FOR i=1 TO n
60    LET t=t/i
70    LET a=a+t
80      PRINT "a("; i; ")="; a
90  NEXT i
100 STOP
```

Exemplu de utilizare:

```
n=?6

a(1)=2
a(2)=2.5
a(3)=2.6666667
a(4)=2.7083333
a(5)=2.7166667
a(6)=2.7180556
```

4.5. Programul 11: FIBO.

Enunţul problemei: Să se calculeze numerele din şirul lui Fibonacci care au indicele în intervalul $[I1,I2]$.

Algoritmul de rezolvare:

Se ştie că şirul numerelor lui Fibonacci verifică relaţia de recurenţă:

$$F(n)=F(n-1)+F(n-2), \text{ pentru } n>1 \quad (4.5.1)$$

şi $F(0)=0$, iar $F(1)=1$.

Pentru obţinerea numerelor care au indicele între $I1$ şi $I2$

trebuie scris un program care va calcula cu ajutorul formulei (4.5.1) toate numerele de la *F(2)* pînă la *F(I2)* şi va afişa doar pe acelea de la *F(I1)* la *F(I2)*. Programul poate fi conceput în două variante care se deosebesc prin modul de păstrare în memoria calculatorului a numerelor calculate. În prima variantă sînt păstrate în memorie toate numerele de la *F(0)* la *F(I2)* şi pentru aceasta este necesară utilizarea instrucţiunii DIM de declarare a tabloului (şirului) *F* cu *I2+1* componente (în calculator indicii încep de la valoarea *1* şi nu *0*). În a doua variantă se vor păstra în memoria calculatorului doar ultimele două numere calculate, *F1* şi *F2*, singurele necesare pentru calculul următorului număr *F*, aşa după cum rezultă din relaţia de recurenţă (4.5.1). Deci în această variantă necesarul de memorie este mai mic, în schimb la fiecare pas trebuie efectuate transferurile *F1*——>*F2* şi *F*——>*F1* pentru ca *F1* şi *F2* să conţină ultimele două valori calculate.

În calculatoarele personale numerele se aproximează la 11 cifre semnificative ceea ce face ca dintre numerele şirului lui Fibonacci să se poată calcula exact doar termenii pînă la F(39), lucru care de altfel va fi precizat de către program în cazul în care I2 > 39.

a. Variabile de intrare:

I1, I2 - indicele de la care şi indicele pînă la care se cer să fie calculate şi afişate valorile numerelor din şirul lui Fibonacci.

b. Variabile de ieşire:

F(I1), F(I1+1), ... , F(I2) în prima variantă şi respectiv valorile succesive ale variabilei F în varianta a doua.

c. Variabile de lucru:

I - contor pentru valorile indicilor termenilor;

F1 şi F2 în a doua variantă conţin valorile termenilor F(I-1), respectiv F(I).

d. Algoritmul propriu-zis:

Varianta întîi:

Date I1,I2;
Fie F(1):=0; F(2):=1.
Pentru I=3,I1 execută F(I):=F(I-1)+F(I-2) sf {pentru}

```
Pentru I=I1+1,I2+1 execută
    Fie F(I):=F(I-1)+F(I-2),
    Tipăreşte F(I)
sf {pentru}
```

Varianta a doua:

```
Date I1,I2;
Fie F2:=0;      F1:=1;
Pentru I=2,I1-1 execută F:=F1+F2;
                        F2:=F1;
                        F1:=F;
sf {pentru}
Pentru I=I1,I2 execută F:=F1+F2;
                       Tipăreşte F;
                       Fie F2:=F1; F1:=F;
sf {pentru}
```

Programul BASIC:

a) Varianta întîi

```
1  REM   Programul 11: FIBO1
2  REM ************************
10 REM * Acest program calcu-
12 REM * leaza numerele lui
14 REM * Fibonacci cu indicele
16 REM * intre I1 si I2
20 REM ************************
60 INPUT "I1=";I1
62 INPUT "I2=";I2
66 IF I1<3 THEN GOTO 200
70 IF I1>I2 OR I1<>INT I1 OR I2<>INT I2 THEN
   GOTO 200
75 CLS
80 IF I2 > 39 THEN PRINT "Valori exacte sint numai
   pina la termenul de indice 39": PRINT
90 DIM F(I2+1)
100 LET F(1)=0
110 LET F(2)=1
120 FOR I=3 TO I1
130 LET F(I)=F(I-1)+F(I-2)
140 NEXT I
```

```
150 FOR I=I1+1 TO I2+1
160 LET F(I)=F(I-1)+F(I-2)
170 PRINT "F(";I-1;")=";F(I)
180 NEXT I
190 STOP
200 PRINT "Valori gresite"
210 PRINT "I1=";I1;"   I2=";I2
220 GOTO 60
```

b) Varianta a doua

```
1  REM   Programul 11A:  FIBO2
10 REM ********************************
20 REM * Acest program calculeaza     *
30 REM * numerele lui Fibonacci cu    *
40 REM * indicele intre I1 si I2      *
50 REM ********************************
60 INPUT "I1=";I1: INPUT "I2=";I2
66 IF I1<3 THEN GOTO 300
70 IF I1>I2 OR I1<>INT I1 OR I2<>INT I2 THEN GO TO 300
75 CLS
80 IF I2 > 39 THEN PRINT "Valori exacte sint numai
   pînă la termenul de indice 39": PRINT
90 LET F2=0
100 LET F1=1
110 FOR I=2 TO I1-1
120 LET F=F1+F2
130 LET F2=F1
140 LET F1=F
150 NEXT I
160 FOR I=I1 TO I2
170 LET F=F1+F2
180 PRINT "F(";I;")=";F
190 LET F2=F1
200 LET F1=F
210 NEXT I
220 STOP
300 PRINT "Valori gresite"
310 PRINT "I1=";I1;"    I2=";I2
320 GOTO 60
```

Exemplu de utilizare:

```
I1=?36
I2=?41
Valori exacte sint numai pina la
termenul de indice 39
F(36)=14930352
F(37)=24157817
F(38)=39088169
F(39)=63245986
F(40)=1.0233416E+8
F(41)=1.6558014E+8
```

4.6. Programul 12: RADICAL.

Enunţul problemei: Să se găsească o valoare aproximativă *r* pentru radical din a, unde *a* este un număr real pozitiv.

Algoritmul de rezolvare:

Se cunoaşte din manualul de analiză pentru clasa a XI-a că şirul definit recurent prin:

$$x(n+1) = (x(n) + a/x(n))/2, \quad n > 0 \qquad (4.6.1)$$

cu *x(1)* ales pozitiv este monoton şi mărginit, iar limita sa este radical de ordin *2* din *a*. Aşadar se poate folosi relaţia (4.6.1) pentru calculul a cît mai multor termeni ai şirului. Calculul termenilor şirului se va face pînă cînd modulul diferenţei a doi termeni consecutivi va fi mai mic decît un număr dat *EP*. După oprirea procesului de calcul ultimul termen calculat va reprezenta valoarea aproximativă a radicalului.

Organizarea memoriei se poate face în două moduri. În prima variantă programul foloseşte pentru fiecare valoare calculată a şirului cîte o celulă de memorie distinctă. Acest mod de lucru necesită folosirea instrucţiunii **DIM** şi cere să cunoaştem (estimăm) o margine pentru numărul de termeni (ai şirului) necesari pentru a obţine aproximaţia dorită. În a doua variantă programul foloseşte doar două celule de memorie în care se vor păstra pe rînd (succesiv) toţi termenii calculaţi ai şirului. În acest caz nu este necesar să cunoaştem apriori numărul de termeni care vor fi calculaţi; se pot folosi doar două celule de memorie (variabile) deoarece într-o celulă *(XV)* se păstrează valoarea ultimului

termen calculat iar în cealaltă (*XN*) noul termen calculat din *XV* cu ajutorul formulei:

$$XN=(XV + a/XV)/2 \qquad (4.6.2)$$

Dacă $|XN-XV| >= EP$, adică nu s-a realizat precizia dorită *EP*, înainte de reluarea procesului de calcul (calculul următorului termen al şirului) este necesar transferul valorii lui *XN* în *XV* deoarece ultimul termen la acest moment are valoarea în celula corespunzătoare lui *XN* şi ea trebuie să fie în celula lui *XV*.

a. Variabile de intrare:

A - valoarea din care dorim să se calculeze radicalul;

EP - aproximaţia cu care dorim să fie calculat acest radical.

b. Variabile de ieşire:

I - precizează numărul de iteraţii executate de către program ;

X(I) pentru primul program respectiv XN pentru al doilea program = valoarea aproximativă a radicalului de ordinul 2 din A.

c. Variabile de lucru:

X - tablou pentru prima variantă;

XV - variabilă simplă pentru varianta a doua;

d. Algoritmul propriu-zis:

Varianta întîi:

```
Date EP,A;
Fie X(1):=A; Fie I:=2;
Repetă X(I):=(X(I-1)+A/X(I-1))/2,
       I:= I+1;
pînă cînd |X(I)-X(I-1)| < EP sf {repetă}
Tipăreşte I,X(I).
```

Varianta a doua:

```
Date A, EP;
Fie XN:=A, I:=1;
Repetă XV:= XN;
       XN:=(XV+A/XV)/2;
       I := I+1;
pînă cînd |XV-XN| < EP sf {repetă}
Tipăreşte I,XN.
```

Programul BASIC:

a) Varianta întîi:

```
1  REM  Programul 12: RADICAL1
10 REM **************************
12 REM * Acest program calcu-
14 REM * leaza o valoare aprox.
16 REM * pentru radicalul de
18 REM * ordin 2 dintr-o valoare
20 REM * data A
22 REM **************************
23 REM
70 DIM X(100)
80 INPUT "A="; A
90 IF A<0 THEN CLS:PRINT"Valoare negativa, dati
      alta ":GO TO 80
100 INPUT "EP="; EP
110 LET X(1)=A
120 LET I=2
130 LET X(I)=(X(I-1)+A/X(I-1))/2
140 IF ABS (X(I)-X(I-1)) < EP THEN GO TO 170
150 LET I=I+1
160 GO TO 130
170 PRINT "Radical de ordin 2 din ";A;" este"
175 PRINT "    r= ";X(I)
180 PRINT "S-au efectuat ";I;" iteratii"
190 STOP
```

b) Varianta a doua:

```
1  REM  Programul 12: RADICAL2
10 REM **************************
12 REM * Acest program calcu-
14 REM * leaza o valoare aprox.
16 REM * pentru radicalul de
18 REM * ordin 2 dintr-o val.
20 REM * data A
22 REM **************************
23 REM
70 LET I=1
80 INPUT  "A="; A
90 IF A < 0 THEN CLS : PRINT "Valoare negativa, dati
                        alta": GO TO 80
100 INPUT "EP=";EP
```

```
110 LET XV=A
120 LET XN= (XV+A/XV)/2
130 IF ABS (XN-XV) < EP THEN GO TO 170
140 LET XV=XN
150 LET I=I+1
160 GO TO 120
170 PRINT "Radical de ordin 2 din ";A;" este:"
175 PRINT "    r="; XN
180 PRINT "S-au efectuat ";I;" iteratii"
190 STOP
```

Exemplu de utilizare:

```
A=?10
EP=?0.01
Radical de ordin 2 din 10 este:
   r=3.1622777
S-au efectuat 5 iteratii
```

Asemănător se poate aproxima și valoarea radicalului de ordinul *n* din *A*. În acest scop se folosește faptul că șirul definit prin: x(1) = A;

x(i) = ((n-1)*x(i-1)+A/x(i-1)↑(n-1))/n, (4.6.3)

este convergent la radical de ordinul *n* din *A*.

Întrucît algoritmul și programul **RADICALN** coincid aproape în întregime cu **RADICAL2**, excepție fiind înlocuirea formulei de recurență (4.6.2) cu formula (4.6.3), dăm în continuare numai programul Basic.

```
1  REM  Programul 12A: RADICALN
10 REM ***************************
12 REM * Acest program calcu-
14 REM * leaza o valoare aprox.
16 REM * pentru radicalul de
18 REM * ordin n dintr-o valoare
20 REM * data A
22 REM ***************************
60 INPUT "n=";n
70 LET I=1
80 INPUT  "A="; A

90 IF A < 0 THEN CLS : PRINT "Valoare negativa, dati
                               alta": GO TO 80
```

```
100 INPUT "EP=";EP
110 LET XV=A
120 LET XN= ((n-1)*XV+A/XV↑(n-1))/n
130 IF ABS (XN-XV) < EP THEN GO TO 170
140 LET XV=XN
150 LET I=I+1
160 GO TO 120
170 CLS
172 PRINT "Radical de ordin n din ";A;" este:"
175 PRINT "   r="; XN
180 PRINT "S-au efectuat ";I;" iteratii"
190 STOP
```

Exemplu de utilizare:

```
n=3
A=10
EP=.001
Radical de ordin 3 din 10 este:
   r=2.1544348
S-au efectuat 7 iteratii
```

4.7. Programul 13: COARDATG.

Enunţul problemei: Ştiind că ecuaţia:

$$f(x) = 0 \qquad (4.7.1)$$

are o rădăcina unică în intervalul $[a,b]$, să se aproximeze această rădăcină.

Algoritmul de rezolvare:

Aflarea unei rădăcini într-un interval dat se poate face prin mai multe metode. Dintre aceste am utilizat metoda tangentei (sau metoda lui Newton) şi metoda coardei. Mai există şi metoda înjumătăţirii intervalului, metoda secţiunii de aur, etc.

Metoda combinată a coardei şi a tangentei se aplică în cazul cînd se ştie că ecuaţia admite o soluţie unică în intervalul (a,b) şi că funcţia este monotonă şi îşi păstrează convexitatea în acest interval (deci f' şi f'' îşi păstrează semnul pe (a,b)).

Metoda tangentei constă în reducerea intervalului la unul dintre intervalele (a,t'), (t',b), unde t' este punctul de intersecţie al axei Ox cu una dintre tangentele la graficul funcţiei în punctele $(a,f(a))$, $(b,f(b))$ (cea care taie axa în interiorul

intervalului (a,b)). Deci t' se obține rezolvînd sistemul:

$$\begin{cases} y-f(t') = f'(t) * (t'-t), \\ y = 0, \end{cases}$$

$(t,f(t))$ fiind punctul în care s-a dus tangenta la grafic.

Metoda coardei constă în reducerea intervalului $[a,b]$ la unul dintre intervalele (a,c'), (c',b), unde c' este abscisa punctului în care coarda determinată de punctele $(a,f(a))$ și $(b,f(b))$ taie axa Ox. Deci c' se obține rezolvînd sistemul:

$$\begin{cases} (y-f(c)) / (f(t)-f(c)) = (x-c) / (t-c) \\ y = 0. \end{cases}$$

Programul propune micșorarea intervalului $[a,b]$ prin folosirea atît a metodei tangentei cît și a metodei coardei. Astfel la fiecare pas mai întîi se aplică metoda tangentei și apoi metoda coardei pe intervalul modificat. Se repetă acest procedeu pînă cînd lungimea intervalului este suficient de mică și atunci se alege ca valoare aproximativă mijlocul intervalului obținut.

Pe tot timpul lucrului, intervalul în care se află rădăcina va avea extremitățile c, respectiv t. Inițial c și t sînt determinate astfel:

Dacă sîntem în cazul fig.4.7.1 atunci c:=a și t:=b
altfel (în cazul fig.4.7.2) c:=b și t:=a.

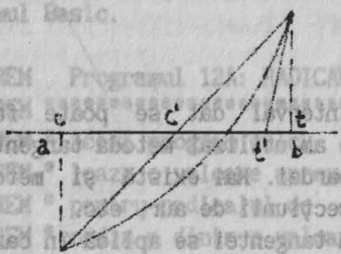


Fig.4.7.1.

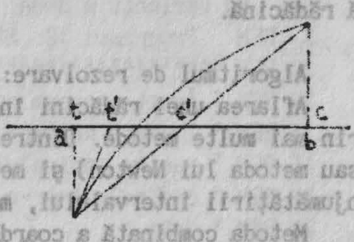


Fig.4.7.2.

Variabile folosite:

a. Variabile de intrare:

A,B - capetele intervalului în care se află rădăcina

ecuaţiei (4.7.1);

EP - conţine precizia cu care se află rădăcina;

expresia funcţiei f se va da ca şi instrucţiune BASIC cu numărul de linie 30;

expresia derivatei funcţiei f se va preciza prin instrucţiunea BASIC cu numărul de linie 40.

expresia derivatei f'' se va preciza prin instrucţiunea cu numărul de linie 50.

b. Variabile de ieşire:

r - valoarea expresiei (a+b)/2 la sfirşitul execuţiei;

c şi t sînt extremităţile intervalului în care se află rădăcina.

c. Variabile de lucru:

F,D,S sînt folosite ca nume pentru valorile funcţiei f şi a derivatelor f', respectiv f''.

d. Algoritmul propriu-zis:

```
Date a,b,EP;
Dacă f(a)=0 atunci Tipăreşte "rădăcina ec. este:",a sf {dacă}
Dacă f(b)=0 atunci Tipăreşte "rădăcina ec. este:",b sf {dacă}
Dacă f(a)*f(b)>0
  atunci Tipăreşte "ecuaţia nu are o singură rădăcină în (a,b)"
        STOP
sf {dacă}
Fie r := (a+b)/2
Dacă f'(r)*f''(r)>0 atunci c:=a; t:=b;
                    altfel c:=b; t:=a
sf {dacă}
Repetă
        t := t - f(t)/f'(t);
        c := c - (t-c) * f(c)/(f(t)-f(c));
        Tipăreşte c,t;
     pînă cînd |c-t| < EP
sf {repetă}
Fie r:=(c+t)/2;
Rezultate r.
```

Programul BASIC:

```
1 REM  Programul 13: COARDATG
2 REM ************************
10 REM * Cu acest program
```

```
11 REM * se afla radacina
12 REM * ecuatiei F(X)=0
13 REM * din intervalul
14 REM * [ a , b ]
15 REM *************************
30 DEF FN F(X)=x↑2-3
40 DEF FN D(X)=2*X
50 DEF FN S(X)=2
100 INPUT "a";a
110 INPUT "b";b
120 IF FN F(a)=0 THEN PRINT "Ecuatia are ca radacina pe",a:STOP
130 IF FN F(b)=0 THEN PRINT "Ecuatia are ca radacina pe",b:STOP
140 INPUT "Precizia EP=";EP
150 IF FNF(a)*FNF(b)>0 THEN
    PRINT "Ec. nu are o singura radacina in (a,b)": STOP
155 LET r=(a+b)/2
160 IF FND(r)*FNS(r) > 0 THEN LET c=a: LET t:=b: GOTO 180
170 LET c=b:  LET t:=a
180 LET t = t - FNF(t)/FND(t)
190 LET c = c - (t-c)*FNF(c) / (FNF(t)-FNF(c))
200 PRINT "c="; c; "   t="; t
210 IF ABS(c-t)>=EP THEN GOTO 180
220 LET r=(c+t)/2
230 PRINT "Radacina este:";r
240 STOP
```

Exemplu de utilizare:

```
a=?1
b=?2
Precizia EP=?.001

c=1.7272727   t=1.75
c=1.7320507   t=1.7321429
Radacina este:1.7320968
```

4.8. Programul 14: TANGENTE.

Enunţul problemei: Se dă un cerc de centru *O* şi rază *R* şi un punct exterior *A*. Să se traseze tangentele din *A* la cercul dat. Se cere să se deseneze figura.

Algoritmul de rezolvare:

Se va folosi faptul că punctele de tangenţă *P* şi *Q* se află la intersecţia cercului dat cu cercul de diametru *AO*. Dacă *(XM,YM)* este centrul acestui cerc şi *R1* este raza sa, atunci punctele *P* şi *Q* se află la intersecţia cercurilor:

$$(X-XO)\uparrow 2 + (Y-YO)\uparrow 2 = R\uparrow 2,$$
$$(X-XM)\uparrow 2 + (Y-YM)\uparrow 2 = R1\uparrow 2 .$$

Prin scădere se obţine ecuaţia axei radicale:

$$(XM-XO)*(2X-XO-XM) + (YM-YO)*(2Y-YO-YM) = R\uparrow 2 - R1\uparrow 2,$$

de unde, pentru $YM \neq YO$, obţinem

$$Y-YO = C1*X + C3$$

unde:

$$C1=\frac{XO-XM}{YM-YO},\quad C2=\frac{R\uparrow 2-R1\uparrow 2}{YM-YO},\quad C3=\frac{C2-C1*(XO+XM)+YM-YO}{2},$$

Înlocuind pe Y-YO în prima ecuaţie obţinem:

$$MA*X\uparrow 2 - 2*MB*X + MC = 0 \qquad (4.8.1)$$

unde:

$$MA = 1 + C1\uparrow 2,$$
$$MB = XO - C1*C3,$$
$$MC = C3\uparrow 2 + XO\uparrow 2 - R\uparrow 2,$$

ecuaţie din care obţinem abscisele punctelor *P* şi *Q*.

Tangentele există numai în cazul cînd *A* este exterior cercului.

a. Variabile de intrare:
   - XO, YO - coordonatele punctului O;
   - XA, YA - coordonatele punctului A;
   - R - raza cercului.

b. Variabile de ieşire:
   - (X1,Y1), (X2,Y2) - coordonatele punctelor de tangenţă.

c. Variabile de lucru:
   - XM, YM, R1 - coordonatele centrului şi raza cercului de diametru AO.
   - C1, C2, C3 - coeficienţi în determinarea axei radicale;
   - MA,MB,MC coeficienţii ecuaţiei (4.8.1);
   - MD - discriminantul ecuaţiei (4.8.1);
   - MR - radical din MD.

d. Algoritmul propriu-zis:

```
Dacă (XA-XO)↑2+(YA-YO)↑2<=R↑2
   atunci tipăreşte "A interior cercului"
   altfel XM:=(XO+XA)/2; YM:=(YO+YA)/2;
      R1:=SQR((XM-XO)↑2+(YM-YO)↑2)
      Dacă YO=YM
         atunci Fie X1:=(XO↑2-XM↑2+R1↑2-R↑2)/[2*(XO-XM)];
                Fie X2:=X1; Fie d:=X1-XO;
                Fie MD:=R↑2-d↑2; MR:= SQR(MD);
                Fie Y1:=YO-MR; Y2:=YO+MR;
         altfel Fie C1 := (XO-XM)/(YM-YO);
                    C2 := (R↑2-R1↑2)/(YO-YM);
                    C3 := (C2-C1*(XO+XM)+YM-YO)/2;
                    MA := 1+C1↑2;
                    MB := XO-C1*C3;
                    MC := C3↑2 +XO↑2 -R↑2;
                    MD := MB↑2 -MA*MC;
                    MR := SQR(MD);
                    X1 :=(MB+MR)/MA; Y1 := YO+C1*X1+C3;
                    X2 :=(MB-MR)/MA; Y2 := YO+C1*X2+C3;
      sf {dacă};
sf {dacă};
```

Programul Basic.

```
1   REM  Programul 16: TANGENTE
2   REM  **************************
10  REM  Din punctul A(XA,YA)
11  REM  se duc tangentele AP si
12  REM  AQ la cercul de centru O
13  REM  si raza r.
14  REM Se presupune ca cercul
15  REM poate fi desenat pe ecran.
16  REM **************************
20  INPUT "XA=";XA
22  INPUT "YA=";YA
24  INPUT "XO=";XO
26  INPUT "YO=";YO
28  INPUT "R="; R
30  LET XM=(XO+XA)/2
32  LET YM=(YO+YA)/2
40  LET D1=XM-XO:  LET D2=YM-YO
44  LET R1=SQR( D1*D1+D2*D2 )
50  PRINT AT 22-YA/8; XA/8; "A"
```

```
60  CIRCLE XO,YO,R
70  PLOT XO,YO
80  PRINT AT 22-YO/8, XO/8; "O"
85  LET MD=(XO-XA)*(XO-XA)+(YO-YA)*(YO-YA)
86  IF MD<=R*R THEN GOTO 500
90  IF YO=YM THEN GOTO 200
100 LET C1=(XO-XM)/(YM-YO)
110 LET C2=(R*R-R1*R1)/(YM-YO)
120 LET C3=C2-C1*(XO+XM)+YM-YO
122 LET C3=C3/2
130 LET MA=1+C1*C1
135 LET MB=XO-C1*C3
140 LET MC=C3*C3+XO*XO-R*R
145 LET MD=MB*MB-MA*MC
150 LET MR=SQR(MD)
160 LET X1=(MB+MR)/MA
165 LET X2=(MB-MR)/MA
170 LET Y1=YO+C1*X1+C3
175 LET Y2=YO+C1*X2+C3
180 GOTO 250
200 LET X1=XO*XO-XM*XM+R1*R1-R*R
205 LET X1=X1/(XO-XM)/2
210 LET X2=X1
220 LET d=X1-XO: LET MD=R*R-d*d
225 LET MR=SQR(MD)
230 LET Y1=YO-MR
235 LET Y2=YO+MR
250 LET A$='P'
260 GOSUB 300
270 LET X1=X2:      LET Y1=Y2
280 LET A$='Q'
285 GOSUB 300
290 STOP
300 REM Se traseaza o tangenta
306 LET xx=X1/8
308 LET yy=22-Y1/8
310 PRINT AT yy,xx; A$
320 PLOT X1,Y1
330 DRAW XA-X1,YA-Y1
340 RETURN
500 PRINT "A este interior cercului"
```

Exemplu de utilizare:

[Se obţine figura dorită. Datele trebuie astfel alese încît să intre figura pe ecran. Programul a fost rulat cu datele:

XA	YA	XO	YO	R
20	20	80	80	40
20	90	180	90	50
200	30	40	70	30
. . .	. . .	. . .	. . .	. . .

]

4.9. Programul 15: GRAFIC.

Enunţul problemei: Să se deseneze graficul funcţiei *F*.

Algoritmul de rezolvare:

În continuare se va indica o metodă de rezolvare a acestei probleme pentru orice funcţie *F* care în intervalul [*a*,*b*] este definită, deci *F* are valori finite în orice punct din acest interval. Pentru a putea întocmi programul presupunem că *F(x)* are valori în intervalul [*c*,*d*] pentru *x* din [*a*,*b*]. In aceste condiţii graficul funcţiei este de forma indicată în fig.4.9.1.

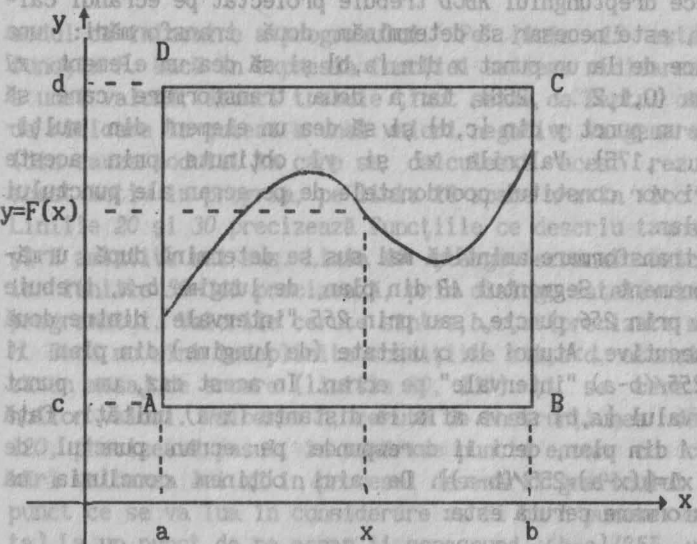
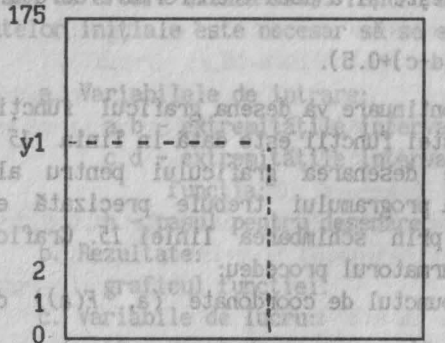


Fig.4.9.1.

Trebuie ca dreptunghiul *ABCD* din plan să fie reprezentat pe ecranul calculatorului, care este o rețea dreptunghiulară de puncte: *256* puncte pe orizontală și *176* puncte pe verticală.



Dacă graficul obținut cu anumite valori pentru *a*,*b*,*c* și *d* nu este satisfăcător (se pierd anumite detalii), atunci se poate cere desenarea graficului cu alte valori pentru aceste variabile. Din această cauză valorile pentru variabilele *a*,*b*,*c*,*d*, vor constitui date de intrare pentru program.

Deoarece dreptunghiul *ABCD* trebuie proiectat pe ecranul calculatorului, este necesar să determinăm două transformări: una care să plece de la un punct *x* din [a,b] și să dea un element *x1* din mulţimea {0,1,2,...,255}, iar a doua transformare care să plece de la un punct *y* din [c,d] și să dea un element din mulţimea {0,1, ... ,175}. Valorile *x1* și *y1* obţinute prin aceste transformări vor constitui coordonatele de pe ecran ale punctului *(x,y)* din plan.

Prima transformare amintită mai sus se determină după următorul raţionament. Segmentul *AB* din plan, de lungime *b-a*, trebuie reprezentat prin *256* puncte, sau prin *255* "intervale" dintre două puncte consecutive. Atunci la o unitate (de lungime) din plan îi corespund *255/(b-a)* "intervale" pe ecran. In acest caz, un punct *x* din intervalul *[a,b]* se va afla la distanţa *(x-a)* unităţi faţă de punctul *A* din plan, deci îi corespunde pe ecran punctul de coordonate x1=[(x-a)*255/(b-a)]. De aici obţinem concluzia că prima transformare cerută este:

G(x)=INT((x-a)*255/(b-a)+0.5),

deoarece vom face o rotunjire a valorii lui *x1* (sub funcţia **INT** ce determină partea întreagă s-a adunat și 0.5). Dacă *x* este în intervalul [a,b], atunci *G(x)* este în mulţimea {0,1,...,255}.

În mod analog se stabilește și a doua transformare cerută:

H(y)=INT((y-c)*175/(d-c)+0.5).

Programul ce va fi dat în continuare va desena graficul funcţiei *F*. Expresia analitică a acestei funcţii este dată in linia *15* a programului. Dacă se dorește desenarea graficului pentru altă funcţie, înainte de execuţia programului trebuie precizată expresia funcţiei respective prin schimbarea liniei *15*. Graficul funcţiei se desenează după urmatorul procedeu:

- se desenează pe ecran punctul de coordonate *(a, F(a))* din plan, după care *x:=a*;

- după ce s-a desenat un punct oarecare, iniţial *(a,F(a))*, valoarea lui *x* se mărește cu un anumit pas, deci *x:=x+h*. Punctul desenat anterior se uneşte cu punctul de coordonate *(x,F(x))* printr-un segment. Această procedură se execută cît timp *x* rămîne în intervalul *[a,b]*. Din cele arătate mai sus se deduce că graficul funcţiei se aproximează printr-o succesiune de segmente.

În continuare se vor face cîteva comentarii cu privire la

modul de întocmire a programului. Pe linia *15* este precizată funcția *F*. Dacă în expresia funcției se cere ridicarea la putere a unei valori, atunci trebuie ținut seama de faptul că încercarea de ridicare la putere a unei valori negative va genera o eroare (din cauza modului în care se calculează acest rezultat). Din acest motiv în program, pe linia 10 apare *x*x* în loc de "*x↑2*". Liniile *20* și *30* precizează funcțiile ce descriu transformările *G* și *H* amintite mai sus. Linia *40* șterge ecranul calculatorului, iar liniile *50-140* precizează, prin dialog, datele inițiale ale programului. Valorile cerute sînt: *a,b,c,d* (precizate mai sus) și *h*. Dacă nu sînt îndeplinite condițiile *a<b,c<d*, atunci se afișează un mesaj de eroare (liniile *80*, *110*) și se cere precizarea noilor valori. Valoarea *h*, cerută de instrucțiunea de la linia *120*, precizează pasul, în număr de puncte ecran cu care se va mări valoarea lui *x1* în procesul desenării graficului (următorul punct ce se va lua în considerare va fi cu *h* puncte spre dreapta). La un punct de pe ecran îi corespund *(b-a)/255* unități din intervalul [*a,b*], deci pasul cu care se va mări *x* va fi *h1=h*(b-a)/255* (valoare calculată la linia *150*). Dacă la desenarea funcției se obține pentru *y=F(x)* o valoare ce nu aparține intervalului [*c,d*] (această condiție se testează în liniile *180* și *260*), atunci desenul nu se va mai trasa și pe ecran apare un mesaj de eroare (linia *380*). Pentru desenarea cu alte valori ale datelor inițiale este necesar să se execute din nou programul.

a. Variabilele de intrare:
   - a,b - extremitățile intervalului pentru valorile lui x;
   - c,d - extremitățile intervalului în care ia valori funcția;
   - h - pasul pentru desenare, în număr puncte ecran.

b. Rezultate:
   - graficul funcției.

c. Variabile de lucru:
   - h1 - pasul de pe axa reală corespunzător celor h puncte ecran;
   - (x,y) - coordonatele punctului curent de pe grafic;
   - (x1,y1) - coordonatele ecran ale punctului (x,y) din plan;
   - (x2,y2) - coordonatele ecran ale ultimului punct desenat;

d. Algoritmul propriu-zis:

1. Citește datele inițiale: a,b,c,d,h;
2. h1:=h*(b-a)/255;
3. Șterge ecranul;
4. x:=a;
5. y:=F(x);
6. x1:=0; y1:=H(y);
   Desenează punctul (x1,y1);
7. x2:=x1; y2:=y1;
8. x:=x+h1;
   Dacă x>b atunci salt la pasul 11;
9. y:=F(x);
10. x1:=G(x); y1:=H(y);
    Desenează segmentul ce unește punctele (x2,y2) și (x1,y1);
    Salt la pasul 7;
11. Dacă a*b<=0 atunci desenează axa Oy;
12. Daca c*d<=0 atunci deseneaza axa Ox;

Programul Basic:

```
1 REM  Programul 15: GRAFIC
10 REM ***************************
11 REM Desenează graficul
12 REM funcției F
13 REM ***************************
14 REM Expresia funcției se dă în linia 15
15 DEF FNF(x)=SIN(x*x)
20 DEF FNG(x)=INT((x-a)*255/(b-a)+0.5)
30 DEF FNH(y)=INT((y-c)*175/(d-c)+0.5)
40 CLS
50 PRINT "Urmează să se precizeze datele inițiale: a,b,c,d:"
60 INPUT "a,b=";a,b
70 IF a<b THEN GOTO 90
80 PRINT "Eroare: trebuie ca a<b" : GOTO 60
90 INPUT "c,d=";c,d
100 IF c<d THEN GOTO 120
110 PRINT "Eroare: trebuie ca c<d" : GOTO 90
120 INPUT "Pasul pentru desenare, în nr.puncte ecran:";h
130 IF h>0 AND h<255 AND h=INT(h) THEN GOTO 150
140 PRINT "Eroare: Pasul trebuie să fie întreg si cuprins
        între 1 și 255." : GOTO 120
150 LET h1=h*(b-a)/255
```

```
160 CLS
165 LET x=a
170 LET y=FN F(x)
180 IF y<c OR y>d THEN GOTO 380
190 LET x1=0
200 LET y1=FN H(y)
210 PLOT x1,y1
220 LET x2=x1: LET y2=y1
230 LET x=x+h1 x1=FN G(x)
280 LET y1= FN H(y)
290 DRAW x1-x2,y1-y2
300 GOTO 220
310 IF a*b>0 THEN GOTO 340
320 LET x=0: LET x1=FN G(x)
330 PLOT x1,0: DRAW 0,175
340 IF c*d>0 THEN GOTO 370
350 LET y=0:  LET y1=FN H(y)
360 PLOT 0,y1 : DRAW 255,0
370 STOP
380  PRINT FLASH 1;"Eroare: Valorile funcţiei depăşesc intervalul
precizat"
390 STOP
```

Exemplu de utilizare:

[Pentru *(a,b)=(-3,3)* şi *(c,d)=(-1,1)* se obţine graficul funcţiei *f* pe intervalul *[a,b]*. Trebuie ca toate valorile lui *f* să se afle în intervalul *[c,d]*.]

4.10. Programul 16: CONICE.

Enunţul problemei: Desenarea unei conice dată prin ecuaţia carteziană implicită.

Algoritmul de desenare:

În continuare se va da o metodă de desenare a oricărei conice (elipsă, hiperbolă, parabolă). Pentru aceasta vom "decupa" din plan un dreptunghi *D={(x,y), a1<= x <=a2, b1<= y <=b2}*, iar punctele ce aparţin conicei şi se află în acest dreptunghi vor fi desenate pe ecranul calculatorului. Programul va cere precizarea dreptunghiului *D* (valorile: *a1,a2,b1,b2*) sau se ia (iniţial) în considerare un dreptunghi *D=[-127,127]x[-87,87]*. Dacă desenul nu

satisface cerinţele (apare prea mic pe ecran, apar prea puţine puncte, etc.) se poate cere desenarea din nou, după redefinirea dreptunghiului *D*.

Este necesar ca dreptunghiul *D* să fie "proiectat" pe ecranul calculatorului. Pentru aceasta trebuie găsite două funcţii (două transformări) *G* şi *H*, care să plece de la punctul *(x,y)* din *D* şi să determine un punct *(x1,y1)* de pe ecranul calculatorului, *x1=G(x)* şi *y1=H(y)*. Aceste transformări au fost determinate la programul precedent şi ele sînt:

```
G(x)=INT((x-a1)*255/(a2-a1)+0.5),
H(y)=INT((y-b1)*175/(b2-b1)+0.5).
```

Ecuaţiile carteziene implicite ale celor trei conice sînt:

a. Pentru elipsă:

$$\frac{x\uparrow 2}{a\uparrow 2} + \frac{y\uparrow 2}{b\uparrow 2} = 1,$$

de unde obţinem:

$$y = b/a\ \sqrt{(a\uparrow 2 - x\uparrow 2)}\ , \qquad y = -\ b/a\ \sqrt{(a\uparrow 2 - x\uparrow 2)},$$

pentru x din [-a,a].

b. Pentru hiperbolă:

$$\frac{x\uparrow 2}{a\uparrow 2} - \frac{y\uparrow 2}{b\uparrow 2} = 1,$$

de unde obţinem:

$$y = b/a\ \sqrt{(x\uparrow 2 - a\uparrow 2)}\ , \qquad y = -\ b/a\ \sqrt{(x\uparrow 2 - a\uparrow 2)},$$

pentru x din [-a,a].

c. Pentru parabolă:

$$y\uparrow 2 = 2px,\ x>=0,\ \ (p>0),$$

de unde se obţine:

$$y = \sqrt{2px}\ , \quad y = -\ \sqrt{2px}\ , \quad x >= 0.$$

Examinînd expresiile obţinute se observă că avem nevoie de urmatoarele trei funcţii:

$$U(x) = b/a\ \sqrt{a\uparrow 2-x\uparrow 2}\ , \ -a <= x <= a; \qquad (4.10.1a)$$

$$V(x) = b/a\ \sqrt{x\uparrow 2-a\uparrow 2}\ , \ x <= -a \text{ sau } x >= a; \qquad (4.10.1b)$$

$$W(x)= \sqrt{2px}\ , \ x >= 0; \qquad (4.10.1c)$$

Pentru a desena ramura superioară a conicei (deasupra axei Ox) vom lua $y=U(x)$, sau $y=V(x)$, sau $y=W(x)$, iar pentru a desena ramura inferioară a conicei (sub axa Ox) se va lua $y=-U(x)$, sau $y=-V(x)$, sau $y=-W(x)$.

Deoarece nu putem desena toate punctele de pe conice (în special la hiperbolă şi parabolă), trebuie să luăm numai punctele din domeniul D, pentru care x este din $[a1,a2]$. Din aceasta cauză variabila x din funcţiile (4.10.1a), (4.10.1b) şi (4.10.1c) va trebui să fie în următoarele intervale:

a. Pentru elipsă:

$$x \in [-a,a] \cap [a1,a2] = [c,d],$$

unde c = max (a1,-a), iar d = min (a2,a);

b. Pentru hiperbolă:

$$x \in ((-\infty,-a] \cup [a,+\infty)) \cap [a1,a2] = [c,d] \cup [c',d'],$$

unde c=a1, d=min (a2,-a), c'=max (a1,a), d'=a2;

c. Pentru parabolă:

$$x \in [0,+\infty) \cup [a1,a2] = [c,d],$$

unde c = max (a1,0), d = a2.

În fiecare din aceste cazuri este posibil ca intervalul (sau intervalele, în cazul hiperbolei) căruia îi aparţine x să fie vid (dacă $c > d$ sau $c' > d'$).

Avînd în vedere asemănările ce există în procesul de desenare, s-a conceput un program general ce abordează problema pentru toate conicele, urmînd ca prin dialog să se precizeze conica dorită. Procesul de desenare este următorul:

1. Se desenează pe ecran punctul ce corespunde la $(c,F(c))$ din plan, unde $F=U$ (pentru elipsă), $F=V$ (pentru hiperbolă), sau $F=W$ (pentru parabolă). Fie $x:=c$.

2. După ce s-a desenat un punct oarecare de pe conică (la început $(c,F(c))$), valoarea lui x se măreşte cu o valoare (pas) h, deci $x:=x+h$. Punctul desenat anterior şi cu noul punct (de coordonate $(x,F(x))$ în plan) se uneşte cu un segment de dreaptă. Acest pas se repetă cît timp x se află în intervalul $[c,d]$.

Pentru hiperbolă se repetă această procedură pentru al doilea interval $[c',d']$.

În acest mod s-a desenat ramura (partea) superioară a conicei (deasupra axei Ox). Pentru desenarea ramurei inferioare a conicei (aflată sub axa Ox) se repetă cele prezentate mai sus, folosind $F=-U$, sau $F=-V$, sau $F=-W$, după natura conicei.

Un punct la care se ajunge în timpul construirii desenului, de coordonate $(x,F(x))$, este posibil să nu aparţină dreptunghiu-

lui *D*, deci segmentul ce unește acest punct cu punctul anterior desenat să nu se poată desena (să nu aparțină ecranului). Aceeași situație se întîlnește și în cazul în care punctul curent, de coordonate *(x,F(x))*, aparține dreptunghiului *D* dar punctul anterior nu a aparținut acestui dreptunghi (această situație se întîlnește numai la desenarea elipsei în cazul în care *b1* > *-b* sau *b2*<*-b*, unde *b1* și *b2* s-au precizat o dată cu dreptunghiul *D*).

a. Variabile de intrare:
   a1, a2, b1, b2 - menționate mai sus.
b. Rezultate:
   desenul cerut.
c. Variabile de lucru:
   r - pentru precizarea ramurii ce se desenează:
      r = 1 - dacă se desenează ramura superioară a conicei (deasupra axei Ox);
      r = 2 - dacă se desenează ramura inferioară a conicei (sub axa Ox);
   t - pentru indicarea conicei ce se desenează:
      t = 1 - dacă se desenează elipsa;
      t = 2 - dacă se desenează hiperbola;
      t = 3 - dacă se desenează parabola;
   k - pentru precizarea cazului în care se află punctul anterior analizat:
      k = 0 - dacă punctul anterior nu s-a putut desena pe ecran, deci el nu se află în dreptunghiul D;
      k = 1 - dacă punctul anterior s-a desenat pe ecran;
   i - pentru precizarea intervalului curent în cazul desenării hiperbolei:
      i = 1 - dacă se ia în considerare intervalul [c,d];
      i = 2 - dacă se în considerare intervalul [c',d'];
   h - pasul pentru modificarea valorii lui x. Dacă acest pas corespunde la j puncte ecran (următorul punct va fi cu j puncte ecran spre dreapta), atunci:

$$h = \frac{a2-a1}{255} * j,$$

   Valoarea lui j se va cere în momentul precizării dreptunghiului D. Implicit, valoarea lui j va fi 4.
d. Algoritmul propriu-zis:

În descrierea algoritmului comentariile vor fi precizate între paranteze drepte.

1. Afişarea pe ecran a variantelor pe care le oferă programul:
   1. Afişarea sau modificarea condiţiilor de lucru;
   2. Desenarea unei elipse;
   3. Desenarea unei hiperbole;
   4. Desenarea unei parabole;
   5. Oprirea programului.
2. Dacă varianta aleasă este 1, atunci se afişează condiţiile de lucru. [Prin condiţiile de lucru se înţelege: descrierea dreptunghiului *D* şi a valorii *j* (ce determină pasul pentru modificarea lui *h*). La început se afişează condiţiile iniţiale de lucru sau cele precizate anterior, după care se întreabă dacă sînt modificări de făcut la aceste condiţii. Dacă se doresc modificări, noile valori sînt cerute, după care se revine din nou la afişarea condiţiilor de lucru. Acest ciclu (afişare, modificare) se repetă pînă ce nu se mai doresc modificări.] Se revine la pasul 1.
3. Dacă se alege varianta a doua, atunci se cer valorile ce determină elipsa.
   t:=1; salt la pasul 7.
4. Dacă se alege varianta a treia, atunci se cer valorile ce determină hiperbola;
   t:=2; salt la pasul 7.
5. Dacă se alege varianta a patra, atunci se cer valorile ce determină parabola;
   t:=3; salt la pasul 7.
6. Dacă se alege varianta a cincea, atunci se opreşte programul.
7. [Precizăm că desenăm ramura superioară] r:=1;
8. [Precizăm valoarea variabilei *i*, utilă dacă desenăm o hiperbolă] i:=1;
9. [Precizarea valorilor *c* şi *d*]
   Dacă t=1 atunci c:=max(a1,-a) şi d:=min(a2,a)
   altfel dacă t=2 atunci c:=a1 şi d:=min(a2,-a)
   altfel c:=max(a1,0) şi d:=a2;
10. [Iniţializarea variabilei *x* şi precizarea faptului că nu a fost desenat un punct anterior]
    x:=c; k:=0;
11. [Verificăm dacă x este în intervalul [c,d]]
    Dacă x > d atunci salt la pasul 18 [Intervalul [c,d] s-a terminat de analizat]
12. [Calculul valorii lui y]
    Dacă t=1 atunci y:=U(x)

altfel dacă t=2 atunci y:=V(x)
    altfel y:=W(x);
Dacă r=2 atunci y:=-y;
13. [Verificăm dacă (x,y) este în dreptunghiul D]
Dacă y < b1 sau y > b2 atunci
k:=0 [punctul nu aparţine lui D] şi salt la pasul 17;
14. [Determinăm coordonatele (x1,y1) de pe ecran pentru (x,y) din dreptunghiul D]
x1:=G(x); y1:=H(y);
15. [Desenăm un punct sau un segment de dreaptă]
Dacă k=0 atunci PLOT(x1,y1) şi k:=1
altfel DRAW(x1-x2,y1-y2);
16. [Păstrăm coordonatele ultimului punct desenat de pe conică]
x2:=x1; y2:=y1;
17. [Luăm în considerare următorul punct]
x:=x+h; salt la pasul 11;
18. [Mărind *x* cu pasul *h* este posibil ca *x* să nu ia valoarea *d* necesară la desenare. Se verifică acest lucru.]
Dacă x<d+h atunci x:=d şi salt la pasul 12.
19. [Dacă desenăm o hiperbolă şi am luat în considerare numai primul interval, atunci trecem la al doilea interval]
Dacă t <> 2 atunci salt la pasul 20 [nu desenăm hiperbola];
Dacă i=2 atunci salt la pasul 20 [Am luat în considerare şi al doilea interval];
c:=max(a1,a); d:=a2; i:=2; [luăm intervalul [c',d']];
Salt la pasul 9;
20. [Dacă nu am desenat şi ramura inferioară, atunci o desenăm]
r:=r+1; dacă r=2 atunci salt la pasul 8;
21. [Conica s-a terminat de desenat]
[Dacă axa *Ox* intersectează dreptunghiul D, ea se desenează]
Dacă b1*b2 > 0 atunci mergi la pasul 22;
h:=0; y1:=H(y);
PLOT(0,y1); DRAW(255,0);
22. [Dacă axa Oy intersectează dreptunghiul D,atunci ea se desenează]
Dacă a1*a2 > 0 atunci salt la pasul 1;
x:=0; x1:=G(x);
PLOT(x1,0); DRAW(0,175);
23 Salt la pasul 1.

Programul BASIC:

```
1 REM  Programul 16: CONICE
2 REM **********************
3 REM    desen
4 REM **********************
5 REM
10 LET a1=-127:  LET a2=127: LET b1=-87: LET b2=87: LET j=4:
   LET h=4
20 DEF FN G(x)=INT((x-a1)*255/(a2-a1)+0.5)
30 DEF FN H(y)=INT((y-b1)*175/(b2-b1)+0.5)
40 DEF FN U(x)=b*SQR(a*a-x*x)/a
50 DEF FN V(x)=b*SQR(x*x-a*a)/a
60 DEF FN W(x)=SQR(2*p*x)
70 CLS: PRINT
80 PRINT "Programul ofera functiile:": PRINT
90 PRINT "  1. Afisarea sau modificarea conditiilor de lucru.":
   PRINT
100 PRINT "  2. Desenarea unei elipse.": PRINT
110 PRINT "  3. Desenarea unei hiperbole.": PRINT
120 PRINT "  4. Desenarea unei parabole.": PRINT
130 PRINT "  5. Oprirea programului.": PRINT
140 PRINT "Ce functie alegeti(1-5)?"
150 LET R$=INKEY$
160 IF R$="1" THEN GO TO 220
170 IF R$="2" THEN GO TO 400
180 IF R$="3" THEN GO TO 500
190 IF R$="4" THEN GO TO 530
200 IF R$="5" THEN STOP
210 GO TO 150
220 CLS:
    PRINT  "Conditiile  de lucru stabilite pina in  acest moment
          sint:"
230 PRINT    "  - pe  ecranul  calculatorului   se   reprezinta
          dreptunghiul din plan pentru care:"
240 PRINT:PRINT "      ";a1;"<=x<=";a2
250 PRINT:PRINT "      ";b1;"<=y<=";b2
260 PRINT:PRINT " - pasul pentru desenarea conicelor, in numar
          puncte ecran, este egal cu:";j
270 INPUT "Doriti modificarea acestor valori(D,N)?";R$
280 IF R$="D" OR R$="d" THEN GO TO 310
```

```
290 IF R$="N" OR R$="n" THEN GO TO 70
300 GO TO 270
310 CLS
315 PRINT "Noile limite pentru x: a1 si a2:"
320 INPUT "a1=";a1,"a2=";a2
330 IF a1>=a2 THEN PRINT "Valori eronate!": GO TO 315
340 PRINT "Noile limite pentru y: b1 si b2:"
350 INPUT "b1=";b1,"b2=";b2
360 IF b1>=b2 THEN PRINT "Valori eronate!": GO TO 340
370 INPUT "Pasul in nr.puncte ecran ";j
380 IF j<=0 THEN PRINT "Valoare eronata!": GO TO 370
390 LET h=(a2-a1)*j/255: GO TO 220
400 CLS:
    LET t=1: LET S$="+"
410 PRINT: PRINT "Ecuatia unei elipse este de forma:":
    PRINT: PRINT
420 PRINT "    2     2 "
430 PRINT "   x     y  "
440 PRINT "  ---- ";S$;" ---- = 1"
450 PRINT "    2     2 "
460 PRINT "   a     b  "
470 PRINT: PRINT:
    PRINT "Care sint valorile pentru a si b?"
480 INPUT "a=";a,"b=";b: GO TO 600
500 CLS:
    LET t=2: LET S$="-"
510 PRINT: PRINT "Ecuatia unei hiperbole este de forma:":
    PRINT: PRINT
520 GO TO 420
530 CLS: LET t=3
540 PRINT: PRINT "Ecuatia unei parabole este de forma:":
    PRINT: PRINT
550 PRINT "    2":  PRINT "   y = 2*p*x"
560 PRINT: PRINT: PRINT "Care este valoarea lui p?"
570 INPUT "p=";p
600 CLS: LET r=1
610 LET i=1
620 IF t<>1 THEN GO TO 670
630 LET c=a1: LET d=a2
640 IF c<-a THEN LET c=-a
650 IF d>a THEN LET d=a
```

```
660 GO TO 730
670 IF t<>2 THEN GO TO 710
680 LET c=a1: LET d=a2
690 IF l>-a THEN LET d=-a
700 GO TO 730
710 LET c=a1: LET d=a2
720 IF c<0 THEN LET c=0
730 LET x=c: LET k=0
740 IF x>d THEN GO TO 850
750 IF t=1 THEN LET y=FN U(x): GO TO 780
760 IF t=2 THEN LET y=FN V(x): GO TO 780
770 LET y=FN W(x)
780 IF r=2 THEN LET y=-y
790 IF y<b1 OR y>b2 THEN LET k=0: GO TO 840
800 LET x1=FN G(x): LET y1=FN H(y)
810 IF k=0 THEN PLOT x1,y1: LET k=1: GO TO 830
820 DRAW x1-x2,y2-y1
830 LET x2=x1: LET y2=y1
840 LET x=x+h: GO TO 740
850 IF x<d+[illegible] THEN LET x=d: GO TO 740
855 IF t<>[illegible] THEN GO TO 900
860 IF i=2 THEN GO TO 900
870 LET c=a1: LET d=a2: LET i=2
880 IF c<a THEN LET c=a
890 GO TO 730
900 LET r=r+1: IF r=2 THEN GO TO 610
910 IF b1*b2>0 THEN GO TO 940
920 LET y=0: LET y1=FN H(y)
930 PLOT 0,y1: DRAW 255,0
940 IF a1*a2>0 THEN GO TO 1000
950 LET x=0: LET x1=FN G(x)
960 PLOT x1,0: DRAW 0,175
1000 INPUT "Pt.continuare tastati orice ";i$: GO TO 70
```

Exemplu de utilizare:

[La comanda RUN apare meniul:]

1. Afisarea sau modificarea
   conditiilor de lucru

2. Desenarea unei elipse
3. Desenarea unei hiperbole
4. Desenarea unei parabole
5. Oprirea programului
Ce functie alegeti (1-5)?

[La cererea utilizatorului, care trebuie să furnizeze parametrii conicei, se desenează una din cele trei conice, sau se execută una din celelalte două funcţii indicate.]

4.11. Programul 17: MEDISP.

Enunţul problemei: Fie *x(1)*, *x(2)*, ..., *x(n)*, *n* numere reale, valori observate în urma unui experiment. Să se calculeze media şi dispersia acestor date.

Algoritmul de rezolvare:

a. Variabile de intrare:
  n = numărul datelor;
  (x(i), i=1,n) = datele observate.

b. Variabile de ieşire:
  m = media;
  d = dispersia;

c. Variabile de lucru:
  i = indice de ciclare;
  m şi d vor fi folosite şi ca variabile de lucru, păstrînd valori intermediare.

d. Algoritmul propriu-zis:

```
Fie m := 0;
Pentru i =1, n execută Fie m := m + x(i) sf {pentru}
Fie m := m/n;
Fie d := 0;
Pentru i = 1, n execută
          Fie d := d + (x(i) - m)↑2;
sf {pentru}
Fie d := d/n;
```

Programul BASIC:

```
1    REM Programul 17:  MEDISP
2    REM ******************************
10   REM Calculul mediei si dispersiei
11   REM ******************************
15   REM ----------- citirea datelor
20   INPUT "n ="; n  : DIM  x(n)
30   FOR i = 1 TO n
40       INPUT "x("; (i); ")="; x(i)
50   NEXT i
55   REM ----------- calculul mediei
60   LET m = 0
70   FOR i = 1 TO n
80       LET m = m + x(i)
90   NEXT i
100  LET  m = m / n
105  REM ----------- calculul dispersiei
110  LET d = 0
120  FOR i = 1 TO n
130      LET d = d + ( x(i) - m ) * ( x(i) - m )
140  NEXT i
150  LET d = d / n
155  REM - tiparirea datelor si a rezultatelor
160  FOR i = 1 TO n
170      PRINT "x("; i; ")="; x(i)
180  NEXT i  : PRINT
190  PRINT "media = ";  m
200  PRINT "dispersia = ";  d
210  STOP
```

Exemplu de utilizare:

```
n = 4
x(1)=?1
x(2)=?2
x(3)=?3
x(4)=?4

x(1)= 1
x(2)= 2
x(3)= 3
x(4)= 4
```

media = 2.5
dispersia = 1.25

4.12. Programul 18: HISTO.

Enunţul problemei: Fie *x(1), x(2), ... , x(n)* date primare. Se cere centralizarea lor, adică stabilirea unor intervale şi calculul frecvenţelor datelor iniţiale în aceste intervale. După centralizare se trasează histograma formată din dreptunghiuri cu intervalele folosite în grupare ca baze şi cu ariile proporţionale cu frecvenţele determinate.

Algoritmul de rezolvare:

a. Variabile de intrare:

```
n                        - numărul datelor primare
x(1), x(2), ... , x(n)   - datele primare
```

b. Variabile de ieşire (rezultate):

```
m                          - numărul intervalelor
A(1), A(2), ... , A(m+1)   - capetele intervalelor
F(1), F(2), ... , F(m)     - frecvenţele în intervale
```

c. Variabile de lucru:

```
m1, m2           - minimul şi maximul dintre x(1),...,x(n)
i, j             - indici de ciclare;
x1, x2, Y, p, D, E - variabile folosite la trasarea
                     dreptunghiurilor
```

d. Algoritmul propriu-zis:

```
Fie m := [1 + 1.442776 ln n];
Fie m1 := x(1), m2 := x(1);
Pentru i = 2, n execută
     Dacă m1 > x(i) atunci Fie m1 := x(i) sf {dacă};
     Dacă m2 < x(i) atunci Fie m2 := x(i) sf {dacă};
sf {pentru}
Dacă m1 = m2 atunci Fie m := 1 sf {dacă}
Fie A(1) := m1, A(m+1) := m2;
Fie p:=(A(m+1)-A(1))/m;
```

```
Pentru i=1,n execută
    Fie F(i):=0; A(i):=m1+(i-1)*p;
sf {pentru}
Dacă p = 0 atunci Fie F(1) := n; STOP Sf {dacă}
Pentru i = 1,m execută
    Fie j := [(x(i)-m1)/p] + 1;
    Dacă j > m atunci Fie j := m sf {dacă};
    Fie F(j) := F(j) + 1;
sf {pentru}
```

Programul BASIC:

```
 1  REM Programul 18:  HISTO
 2  REM ******************************
10  REM Centralizare + histograma
11  REM ******************************
15  REM ---------- citirea datelor
20  INPUT "Numarul datelor:"; n
30  DIM x(n)
40  FOR i=1 TO n
50      INPUT "x("; (i); ")="; x(i)
60  NEXT i
65  REM ---------- calculul minimului si maximului
70  LET m = INT (1+1.442726 * LN n) : DIM A(m+1): DIM F(m)
80  LET m1 = x(1) : LET m2 = x(1)
90  IF n = 1 THEN GOTO 520
100 FOR i = 2 TO n
110     IF m1 > x(i) THEN LET m1 = x(i)
120     IF m2 < x(i) THEN LET m2 = x(i)
130 NEXT i
140 IF m1 = m2 THEN LET m = 1
150 LET A(1) = m1 : LET A(m+1) = m2
160 LET p = ( A(m+1) - A(1) ) / m
170 FOR i = 1 TO m
180     LET F(i) = 0
190     LET A(i) = m1 + (i-1) * p
200 NEXT i
210 IF p = 0 THEN LET  F(1) = n : GOTO  270
220 FOR i = 1 TO n
230     LET j = INT ( (x(i)-m1)/p) + 1
240     IF j > m THEN LET j = m
250     LET F(j) = F(j) + 1
260 NEXT i
```

```
265 REM ---------- tiparire intervale + frecvente
270 PRINT "Intervalul"; TAB 20 ; "Frecventa" : PRINT
280 FOR i = 1 TO m-1
290     PRINT "["; A(i); ","; A(i+1); ")"; TAB 20; F(i)
300 NEXT i
310 PRINT "["; A(m); ","; A(m+1); "]"; TAB 20; F(m)
313 PRINT "Apasati orice tasta pentru continuare !"
314 IF INKEY$="" THEN GOTO 314
315 REM ---------- trasare histograma
320 CLS:  LET D = F(1)
330 IF m = 1 THEN  GOTO 370
340 FOR i = 2 TO m
350     IF D < F(i) THEN LET D = F(i)
360 NEXT i
370 LET x2 = 4 : LET E = A(m+1)-A(1)
380 PLOT 0,10 : DRAW 255,0
390 FOR i = 1 TO m
400   LET x1 = x2 : LET x2 = 250
410   IF E<>0 THEN  LET x2 = x1 + INT ((A(i+1)-A(i))*245/E+0.5)
420   LET Y = 10 + INT (F(i)*160/D + 0.5)
430   PLOT x1 , 5  : DRAW 0 , Y-5
440   DRAW x2-x1 , 0 : DRAW 0 , 10-Y
450   IF F(i) = 0 THEN  GOTO 500
460   LET j = x1 + 1
470   IF j >= x2  THEN  GOTO 500
480   PLOT j , 10 : DRAW 0 , Y-10
490   LET j=j+1  : GOTO 470
500 NEXT i
510 STOP
520 PRINT "Prea putine date": GOTO 20
```

Exemplu de utilizare:

```
        . . .     [Se introduc datele: 1, 3, 3, 4.]

Numarul datelor: 4
x(1) = 1
x(2) = 3
x(3) = 3
x(4) = 4
```

```
Intervalul  Frecventa
[1 , 2)     1
[2 , 3)     0
[3 , 4]     3

Apasati orice tasta pt. continuare !

          !!!!!!!!
          !!!!!!!!
          !!!!!!!!
          !!!!!!!!
!!!!!!!   !!!!!!!!
!!!!!!!   !!!!!!!!
-----------------------
```

4.13. Programul 19: MEDII.

Enunţul problemei: Să se ordoneze elevii unei clase după mediile generale.

Algoritmul de rezolvare:

a. Variabile de intrare:
   M - numărul materiilor la care elevii au note;
   NR - numărul elevilor clasei;
   E$(i) - numele elevului "i";
   N(i,j) - nota elevului "i" la disciplina "j".
b. Variabile de ieşire:
   v(i) - media generala a elevului "i";
   Variabilele de mai sus. Reprezintă aceleaşi informaţii dar în altă ordine. Ca rezultat se tipăresc numele elevilor şi media lor generală, în ordinea descrescătoare a mediilor.
c. Variabile de lucru:
   i,j - variabile de ciclare;
   Inv - număr de inversiuni, folosit la ordonarea descrescătoare.
d. Algoritmul propriu-zis.

Algoritmul ORDONARE este:
Date n, m, (N(i), (M(i,j), j=1,m), i=1,n);
Pentru i = 1,n execută

```
    Calculează media generală v(i) a elevului "i";
sf {pentru}
Repetă Inv := 0;
    Pentru i = 1,n-1 execută
        Daca v(i) < v(i+1) atunci
            Inv := Inv + 1;
            Schimbă între ele informaţiile elevilor "i" şi "i+1"
        sf {dacă}
    sf {pentru}
pînăcînd Inv = 0 sf {repetă}
```

Programul Basic:

```
  1  REM Programul 19: MEDII
  2  REM *********************
 10  REM Se ordoneaza elevii
 11  REM unei clase in ordi-
 12  REM nea descrescatoare
 13  REM a mediilor generale
 14  REM *********************
 18  REM M = nr.materiilor
 19  REM NR = nr.elevilor
 20  REM E$(i) - numele elevului "i"
 21  REM N(i,j) - nota elevului i
 22  REM    la disciplina j
 25  READ NR,M
 30  DIM E$(NR,20):   DIM N(NR,M)
 33  DIM V(NR)
 40  PRINT "Numele si prenumele   Media"
 50  FOR i=1 TO NR
 55      LET s=0
 60      READ E$(i)
 65      FOR j=1 TO M
 70          READ N(i,j)
 80          LET s=s+N(i,j)
 90      NEXT j
100      LET v(i)=s/M
110      PRINT E$(i);
120      PRINT " "; v(i)
130  NEXT i
190  LET INV=0
```

```
200  FOR i = 1 TO NR-1
210       IF v(i)>=v(i+1) THEN GOTO 250
220       LET T=v(i)
221       LET v(i)=v(i+1)
222       LET v(i+1)=T
230       LET T$=E$(i)
231       LET E$(i)=E$(i+1)
232       LET E$(i+1)=T$
240       LET INV=INV+1
250  NEXT i
260  IF INV<>0 THEN GOTO 190
270  PRINT: PRINT
300  FOR i=1 TO NR
310       PRINT E$(i);" ";v(i)
320  NEXT i
400  DATA 3,14
410  DATA 'Abrudan Adela'
411  DATA 9,8,5,8,7,7,6,7,10,9
412  DATA 8,9,10,10
420  DATA 'Florea Sorin'
421  DATA 8,6,9,5,8,7,10,8,8,9
422  DATA 9,8,10,10
430  DATA 'Popescu Alin'
431  DATA 8,8,9,9,8,7,10,8,8,9
432  DATA 9,8,10,10
440  STOP
```

Exemplu de utilizare:

```
Numele si prenumele  Media
Abrudan Adela   8.0714286
Florea Sorin   8.2142857
Popescu Alin   8.6428571

Popescu Alin   8.6428571
Florea Sorin   8.2142857
Abrudan Adela   8.0714286
```

5. PRODUSE PROGRAM

Spre deosebire de secţiunile precedente, în acest capitol vom prezenta cîteva programe Basic mai ample. Ele au în vedere tot programarea unor algoritmi de rezolvare a unor probleme de matematică din liceu. Complexitatea acestor algoritmi împiedică însă predarea lor la clasă în detaliu. Profesorul poate însă să-i prezinte în ansamblu, după care să detalieze părţi din program. Acest lucru este uşurat de prezentarea unor tabele în care se descriu părţile funcţionale ale programelor. De asemenea, sînt date textele complete ale majorităţii programelor.

Programele pot fi însă utilizate în scop didactic şi fără a cunoaşte descrierea lor Basic. Prin "meniuri" explicative, utilizatorului îi sînt oferite posibilităţile de utilizare şi modul de activare al acestor programe.

5.1. Programul 19: RESTURI.

Enunţul problemei: Programul de faţă are scopul de a familiariza elevii cu operaţiile în clase de resturi şi modul lor de programare în BASIC.

Funcţionarea programului.

Se dă un număr natural *n*, *n* > 1. Dacă pentru *n* se dă valoarea *0* atunci programul se termină.

Pentru un *n* fixat, programul afişează un meniu cu şase posibilităţi de operare în Zn (dacă se tastează una dintre cifrele de la 1 la 6) sau schimbarea valorii lui *n* (tastare 0).

Cele şase tipuri de operaţii sînt:

1. Se dă un număr întreg *i* şi se tipăreşte clasa lui *i* în Zn.
2. Se dau două clase *i* şi *j* din Zn şi se calculează *i+j* şi *i*j* în Zn.
3. Se dă o clasă *i* din Zn şi se tipăreşte clasa opusă (simetricul faţă de adunare) şi clasa inversă (simetricul faţă de înmulţire) dacă aceasta există.
4. Se tipăresc perechile de divizori ai lui zero din Zn, dacă aceştia există.
5. Se tipăreşte tabla adunării modulo *n*.
6. Se tipăreşte tabla înmulţirii modulo *n*.

După alegerea variantei se cere, dacă este cazul, valoarea

lui *i* şi eventual a lui *j*, după care se afişează rezultatele. Acestea rămîn pe ecran pînă la apăsarea unei taste, după care se afişează meniul principal. După dorinţă, se poate alege un alt tip de operaţii din cele şase oferite, sau se poate trece la furnizarea unei alte valori pentru *n*.

Structura programului.

Etichetarea instrucţiunilor este făcută în aşa fel încît să permită evidenţierea părţilor de program care execută diversele tipuri de operaţii dorite. Dacă utilizatorul doreşte, el poate să păstreze numai unele tipuri de operaţii, ştergînd secvenţele de instrucţiuni corespunzătoare celorlalte.

Tabelul de mai jos descrie fiecare secvenţă de program.

Eticheta de început	Descrierea sumară a secvenţei
100	Cere o valoare pentru *n* şi-i verifică corectitudinea
300	Afişează meniul principal şi cere una dintre variantele 0, sau 1,...,6 descrise mai sus.
600	In funcţie de valoarea furnizată, se trimite fie la 100 (cazul 0), fie la una din secvenţele care urmează (cazul 1,...,6).
1000	Tratează cazul 1
1300	Tratează cazul 2
1600	Tratează cazul 3
1900	Tratează cazul 4
2200	Tratează cazul 5
2500	Tratează cazul 6
4000	Aşteaptă apăsarea unei taste, după care trece la secvenţa 300.

Programul BASIC

```
1    REM Programul 19: RESTURI
2    REM *********************
10   REM Operatii in Zn
12   REM *********************
99   REM
100  CLS : PRINT "Dati un intreg n, n>1"
101  PRINT "sau 0 pentru STOP"
110  INPUT "n = ?";n
120  IF n=0 THEN STOP
130  IF n<=1 OR n<>INT n THEN GOTO 100
299  REM
300  CLS: PRINT "Va oferim citeva posibilitati de operare in Z";n
301  PRINT
330  PRINT " 1. Se da i din Z si se obtine
               clasa lui i din Z";n
320  PRINT " 2. Se dau i si j din Z";n;" si se
               calculeaza i+j si  i*j in Z";n
330  PRINT " 3. Se da i din Z";n;" si se obtine
               -i si i↑(-1) in Z";n
340  PRINT " 4. Se tiparesc perechile de
               divizori ai lui 0 in Z";n
350  PRINT " 5. Se tipareste tabla adunarii
                    in Z";n
360  PRINT " 6. Se tipareste tabla inmultirii
                    in Z";n
370  PRINT " 0. Se cere o noua valoare
                  pentru n"
380  INPUT "Ce doriti (1,2,3,4,5,6,0) ?;v
382  PRINT
599  REM
600  IF v=0 THEN GO TO 100
610  IF v=1 THEN GO TO 1000
620  IF v=2 THEN GO TO 1300
630  IF v=3 THEN GO TO 1600
640  IF v=4 THEN GO TO 1900
650  IF v=5 THEN GO TO 2200
660  IF v=6 THEN GO TO 2500
670  GO TO 300
1000 INPUT "i = ?";i
```

```
1010 IF i<>INT i THEN GO TO 1000
1020 LET c=ABS i-INT (ABS (i)/n)*n: IF i<0 THEN LET c=n-c
1030 PRINT i;" (mod ";n;") = ";c
1040 GO TO 4000
1299 REM
1300 INPUT "i = ?";i: INPUT "j = ?";j
1310 IF i<0 OR i>=n OR i<>INT i OR j<0 OR j>=n OR j<>INT j
     THEN GO TO 1300
1320 PRINT i;" + ";j;" (mod ";n;") = ";i+j-INT ((i+j)/n)*n
1330 PRINT i;" * ";j;" (mod ";n;") = ";i*j-INT ((i*j)/n)*n
1340 GO TO 4000
1599 REM
1600 INPUT "i = ?";i
1610 IF i<0 OR i>=n OR i<>INT i THEN GO TO 1600
1620 PRINT "Elementul opus lui ";i;" in Z";n:
     PRINT "este "; n-i
1630 PRINT "Elementul invers lui ";i;" in Z";n;" ";
1640 FOR j=1 TO n-1
1650    LET c=i*j-INT ((i*j)/n)*n
1660    IF c<>1 THEN GO TO 1690
1670    PRINT "este ";j
1680    GO TO 4000
1690 NEXT j
1700 PRINT "nu exista"
1710 GO TO 4000
1899 REM
1900 PRINT "Divizori ai lui 0 in Z";n
1910 LET k=0
1920 FOR i=1 TO n-1
1930    LET c=i*i-INT ((i*i)/n)*n
1940    IF c<>0 THEN GO TO 1960
1950    LET k=k+1: PRINT i;" * ";i;" = 0"
1960    FOR j=i+1 TO n-1
1970       LET c=i*j-INT ((i*j)/n)*n
1980       IF c<>0 THEN GO TO 2000
1990       LET k=k+1: PRINT i;" * ";j;" = 0 si ";j;" * ";i;" = 0"
2000    NEXT j
2010 NEXT i
2020 IF k=0 THEN PRINT "nu exista"
2030 GO TO 4000
2199 REM
```

```
2200 PRINT "Tabla adunarii in Z";n: PRINT " +";
2210 FOR j=0 TO n-1: PRINT j;" ";: NEXT j: PRINT
2220 FOR i=0 TO n-1
2230    PRINT: PRINT i;"   ";
2240    FOR j=0 TO n-1
2250       PRINT i+j-INT ((i+j)/n)*n;" ";
2260    NEXT j
2270 NEXT i
2280 GO TO 4000
2500 PRINT "Tabla inmultirii in Z";n
2510 FOR j=0 TO n-1: PRINT j;" ";: NEXT j: PRINT
2520 FOR i=0 TO n-1
2530 PRINT: PRINT i;"   ";
2540 FOR j=0 TO n-1
2550 PRINT i*j-INT ((i*j)/n)*n;" ";
2560 NEXT j
2570 NEXT i
2580 GO TO 4000
3999 REM
4000 PRINT "Apasati orice tasta"
4005 PAUSE 0
4010 IF INKEY$="" THEN GO TO 4010
4020 GO TO 300
```

Exemple de utilizare:

```
Dati un intreg n, n > 1
sau 0 pentru STOP
n = ?3
Va oferim citeva posibilitati de operare in Z3

    1. Se da i din Z si se obtine
       clasa lui i din Z3
    2. Se dau i si j din Z3 si se
       calculeaza i+j si i*j in Z3
    3. Se da i din Z3 si se obtine
       -i si i↑(-1) in Z3
    4. Se tiparesc perechile de
       divizori ai lui 0 in Z3
    5. Se tipareste tabla adunarii
       in Z3
```

6. Se tipareste tabla inmultirii
   in Z3
0. Se cere o noua valoare
   pentru n

```
Ce doriti (1,2,3,4,5,6,0)?1

i = ?7
7 (mod 3) = 1
```

[Se așteaptă apăsarea unei taste. După apăsarea unei taste, se tipărește din nou meniul principal. Se cere varianta 1, tastînd 1 și urmează:]

```
i = ?-7
-7 (mod 3) = 2
```

[Se așteaptă apăsarea unei taste.După o tastare, reapare meniul; se dă apoi varianta 2 și urmează:]

```
i = ?1
j = ?2
1 + 2 (mod 3) = 0
1 * 2 (mod 3) = 2
```

[Se așteaptă apăsarea unei taste. După o tastare, reapare meniul; se dă apoi varianta 3 și urmează:]

```
i = ?2
Elementul opus lui 2 in Z3
este 1
Elementul invers lui 2 in Z3
este 2
```

[Se așteaptă apăsarea unei taste. După o tastare, reapare meniul: se dă apoi varianta 4 și urmează:]

```
Divizori ai lui 0 in Z3
nu exista
```

[Se așteaptă apăsarea unei taste. După o tastare, reapare meniul; se dă apoi varianta 5 și urmează:]

```
Tabla adunarii in Z3
 + 0 1 2

0  0 1 2
1  1 2 0
2  2 0 1
```

[Se așteaptă apăsarea unei taste. După o tastare, reapare meniul; se dă apoi varianta 6 și urmează:]

```
Tabla inmultirii in Z3
 * 0 1 2

0  0 0 0
1  0 1 2
2  0 2 1
```

[Se așteaptă apăsarea unei taste. După o tastare, reapare meniul; se dă apoi varianta 0 și urmează:]

```
Dati un intreg n, n > 1
sau 0 pentru STOP
n = ?8
```

[Se așteaptă apăsarea unei taste. După o tastare, reapare meniul; se dă apoi varianta 3 și urmează:]

```
i = ?4
Elementul opus lui 4 in Z8
este 4
Elementul invers lui 4 in Z8
nu exista
```

[Se așteaptă din nou apăsarea unei taste. După o tastare, reapare meniul; se dă apoi varianta 4 și urmează:]

```
Divizori ai lui 0 in Z8
2 * 4 = 0 si 4 * 2 = 0
4 * 4 = 0
4 * 6 = 0 si 6 * 4 = 0
```

[Se așteaptă apăsarea unei taste. După o tastare, reapare meniul; se dă apoi varianta 0 și urmează:]

```
Dati un intreg n, n > 1
sau 0 pentru STOP
n = ?0
```

5.2. Programul 20: RIEMANN

Enunţul problemei: Programul **RIEMANN** are scopul de a ajuta elevii să înţeleagă noţiunile de diviziune a unui interval, norma diviziunii, suma Riemann şi convergenţa sumelor Riemann. Programul permite două posibilităţi de alegere a unei diviziuni, iar pentru o diviziune aleasă sînt permise două posibilităţi de alegere a sistemului de puncte intermediare.

Pentru comoditatea programării convenim să descriem elementele sumei Riemann astfel:

a) Pentru un interval $[a,b]$ cu $a < b$, programul RIEMANN consideră ca diviziune un sistem de noduri $(x(i))$:

$$a = x(1),\ x(2),\ \dots,\ x(n),\ x(n+1) = b,$$

unde $n < 500$.

b) Un sistem de puncte intermediare $(p(i))$:

$$p(i) \text{ din intervalul } [x(i),\ x(i+1)],\ i=1,n;$$

c) O sumă Riemann este:

$$R = \sum_{i=1}^{n} f(p(i))\ (x(i+1)-x(i)),$$

unde $f:[a,b] \longrightarrow R$.

Funcţionarea programului.

Mai intîi se cere intervalul $[a,b]$. Dacă se furnizeaza a şi b astfel incît $a > b$, atunci cererea se repetă. Dacă se dă $a = 0$ şi $b = 0$, atunci programul se opreşte.

Urmează definirea diviziunii. În mod automat se pune:

$$x(1) = a.$$

Pentru nodurile următoare există două posibilităţi:

1) alegerea lor de către utilizator, nod cu nod;

2) definirea unui sistem de noduri echidistante.

În varianta *1*, utilizatorului îi sînt cerute nodurile unul după altul, furnizînd şi intervalul în care trebuie să apară. Dacă nodul nu se dă în intervalul specificat, atunci cererea se repetă. Furnizarea se termină cînd se dă ca şi nod valoarea lui b.

În varianta *2* există două posibilităţi de alegere:

3) furnizînd numărul de subintervale, n;

4) furnizînd lungimea unui subinterval oarecare h.

În cazul *3*, toate subintervalele vor avea lungimea:

$$h = (b-a)/n\ .$$

În cazul 4,

$$n = [(b-a)/h],$$

toate intervalele vor avea lungimea h, cu excepţia ultimului care va avea o lungime cel mult h.

După definirea diviziunii, utilizatorul este invitat să-şi aleagă un sistem de puncte intermediare. Pentru aceasta, el are la dispoziţie două variante:

5) alegerea, după dorinţă, a fiecărui $p(i)$ din $[x(i),x(i+1)]$;

6) furnizarea unei expresii $P(i)$ de calcul al punctului $p(i)$ din $[x(i),x(i+1)]$. În varianta 6, programul nu verifică dacă $P(i)$ este corectă sintactic. Dacă $P(i)$ se poate evalua, atunci programul determină $p(i)$ astfel:

$$p(i) = \begin{cases} x(i), & \text{dacă } P(i) < x(i), \\ x(i+1), & \text{dacă } P(i) > x(i+1), \\ P(i), & \text{în rest.} \end{cases}$$

După definirea sistemului de puncte intermediare se cere expresia $f(x)$.

În sfirşit, programul tipăreşte ca şi rezultate intervalele diviziunii, punctele intermediare, norma diviziunii şi valoarea sumei Riemann. Aceste rezultate rămîn pe ecran pînă la apăsarea unei taste, după care programul este reluat de la început, cerînd alte valori pentru a şi b.

Structura programului.

Etichetarea instrucţiunilor este făcută în aşa fel încît să permită evidenţierea părţilor de program care execută diversele variante cerute. Dacă utilizatorul doreşte, poate să reducă fie modalităţile de definire a diviziunii, fie cele de alegere a punctelor intermediare. Pentru aceasta este suficient să şteargă secvenţele de program care nu-i sînt necesare.

Tabelul care urmează descrie rolul fiecărei secvenţe din program.

Eticheta de început	Descrierea sumară a secvenţei
10	Declaraţiile; citirea şi testarea valorilor pentru *a* şi *b*.
100	Cere variantele *1* sau *2* pentru alegerea nodurilor diviziunii
200	Se cer (varianta *1*) nodurile *x(2)*,*x(3)*,... pînă cînd se dă un nod = *b*. Apoi se trece la cererea punctelor intermediare (secventa 1000).
300	Cere una dintre variantele *3* sau *4* pentru definirea diviziunii echidistante.
400	Determină *h* în varianta *3*.
500	Determină *n* în varianta *4*.
600	Determină nodurile *x(i)* în variantele *3* sau *4*.
1000	Cere una dintre variantele *5* sau *6* de alegere a punctelor intermediare *p(i)*
1200	Cere cele *n* puncte intermediare *p(i)*
1400	Cere funcţia *P(i)* şi calculează punctele *p(i)*
2000	Calculează norma diviziunii,suma Riemann şi tipăreşte intervalele de diviziune şi punctele intermediare
2500	Tipăreşte valoarea normei şi a sumei Riemann; aşteaptă o tastare pentru a relua de la început.

```
      Programul BASIC
1     REM Programul 20: RIEMANN
2     REM ***********************
3     REM Pentru [a,b] dat si
4     REM functia f data
5     REM se calculeaza
6     REM suma Riemann
7     REM ***********************
8     REM
10    LET Maxn=500
20    DIM x(Maxn): DIM p(Maxn)
30    CLS: PRINT "Se da un interval de numere
                reale [a,b]":
      PRINT "(Daca a=0 si b=0 atunci STOP)"
40    INPUT "a = ?";a
50    INPUT "b = ?";b
60    IF a=0 AND b=0 THEN STOP
70    IF a>=b THEN GO TO 30
80    LET x(1)=a
99    REM
100   CLS: PRINT "Va oferim doua moduri de definire
                a unei diviziuni peste" :
          PRINT  "[";a;" , ";b;"]"
110   PRINT " 1. Alegerea de catre Dv. a
                nodurilor intermediare"
120   PRINT " 2. Definirea unei diviziuni
                echidistante"
130   INPUT "Care varianta o alegeti (1,2) ?";v
140   IF v<>1 AND v<>2 THEN GO TO 100
199   REM
200   IF v=2 THEN GO TO 300
210   LET n=1: LET xs=a
220   INPUT "x(";(n+1);") din (";(xs);" , ";(b);"] = ?";x(n+1)
230   IF x(n+1)=b THEN GO TO 1000
240   IF x(n+1)<=xs OR x(n+1)>b THEN GO TO 220
250   LET n=n+1: LET xs=x(n): IF n>Maxn-1 THEN GO TO 1000
260   GO TO 220
299   REM
300   PRINT " 3. Prin numarul de subintervale"
310   PRINT " 4. Prin lungimea unui subinterval"
320   INPUT "Care varianta o alegeti (3,4) ?";v
```

```
330  IF v<>3 AND v<>4 THEN GO TO 300
399  REM
400  IF v=4 THEN GO TO 500
410  INPUT "n = ?";n
420  IF n<1 OR n>=Maxn OR n<>INT n THEN GO TO 410
430  LET h=(b-a)/n: GO TO 600
499  REM
500  INPUT "h = ?";h
510  IF h<=0 OR h>=b-a THEN GO TO 500
520  LET n=INT ((b-a)/h)+1: IF n>=Maxn THEN GO TO 500
599  REM
600  FOR i=2 TO n: LET x(i)=a+(i-1)*h: NEXT i
610  LET x(n+1)=b
999  PRINT
1000 CLS: PRINT "In diviziune se poate alege un
              sistem de puncte  intermediare:"
1010 PRINT " 5. Alegind Dv. fiecare punct"
1020 PRINT " 6. Furnizind o explesie P(i)
            de calcul al punctului din al
            i-lea subinterval":
     PRINT "    [x(i),x(i+1)]"
1030 INPUT "Care varianta o alegeti (5,6) ?";v
1040 IF v<>5 AND v<>6 THEN GO TO 1000
1199 REM
1200 IF v=6 THEN GO TO 1400
1210 FOR i=1 TO n
1220   INPUT"p(";(i);") din [";(x(i));",";(x(i+1));"]=?";p(i)
1230   IF x(i)>p(i) OR p(i)>x(i+1) THEN GO TO 1220
1240 NEXT i
1250 GO TO 2000
1399 REM
1400 INPUT "P(i) = ";s$
1410 DEF FN s(i)=VAL s$
1420 FOR i=1 TO n
1430   LET p(i)=FN s(i)
1440   IF p(i)<x(i) THEN LET p(i)=x(i)
1450   IF p(i)>x(i+1) THEN LET p(i)=x(i+1)
1460 NEXT i
1999 REM
2000 INPUT "f(x) = ?";f$
2010 DEF FN f(x)=VAL f$
```

```
2020 LET norma=0: LET suma=0
2030 CLS: PRINT "Pentru intervalul [";a;" , ";b;"] si"
2031 PRINT  "f(x)=";f$:
     PRINT " avem diviziunea si punctele:"
2032 PRINT
2040 FOR i=1 TO n
2050    PRINT " [";x(i);" , ";x(i+1);"] contine ";p(i)
2060    LET hi=x(i+1)-x(i): IF hi>norma THEN LET norma=hi
2070    LET suma=suma+FN f(p(i))*hi
2080 NEXT i
2500 PRINT "Norma diviziunii este ";norma
2510 PRINT "Valoarea sumei Riemman este "
2511 PRINT "    suma ="; suma
2520 IF INKEY$="" THEN GO TO 2520
2530 GO TO 30
```

Exemplu de folosire:

```
Se da un interval de numere
     reale [a,b]
 (Daca a=0 si b=0 atunci STOP)
a = ?0
b = ?1
Va oferim doua moduri de
definire a unei diviziuni peste
 [0,1]
 1. Alegerea de catre Dv. a
      nodurilor intermediare
 2. Definirea unei diviziuni
      echidistante

Care varianta o alegeti (1,2)?1
x(2) din (0,1]=?0.4
x(3) din (0.4,1]=?0.7
x(4) din (0.7,1]=?0.9
x(5) din (0.9,1]=?1

In diviziune se poate alege un
sistem de puncte intermediare
5. Alegind Dv. fiecare punct
6. Furnizind o expresie P(i)
```

```
de calcul al punctului din al
i-lea subinterval
    [x(i),x(i+1)]
Care varianta o alegeti (5,6) ? 6
P(i) = "(x(i) + x(i+1)) / 2"

f(x) = "x"

Pentru intervalul [0,1] si
     f(x) = x
avem diviziunea si punctele:
[0,0.4] contine 0.2
[0.4,0.7] contine 0.55
[0.7,0.9] contine 0.8
[0.9,1] contine 0.95

Norma diviziunii este 0.4
Valoarea sumei Riemann este
     suma = 0.5
```

[Rezultatele rămîn pe ecran pînă la apăsarea unei taste. Presupunem că după reluare am furnizat aceleași valori pentru a și b, varianta 2 la alegerea diviziunii, în cadrul acesteia am dorit varianta 3 pentru n = 4. Urmează:]

```
Care  varianta o alegeti (5,6)?6
P(i) = "x(i)*(i-1)/n+x(i+1)*(n-i+1)/n".

f(x) ="x"
Pentru intervalul [0,1] si
    f(x)=x
avem diviziunea si punctele:
[0,0.25] contine 0.25
[0.25,0.5] contine 0.4375
[0.5,0.75] contine 0.625
[0.75,1] contine 0.8125
Norma diviziunii este 0.25
Valoarea sumei Riemann este
    suma = 0.53125
```

5.3. Programul 21: APROXINT

Enunţul problemei: În programul de faţă vom exemplifica două moduri de calcul aproximativ al integralei

$$I = \int_a^b f(t)\,dt$$

şi anume:

- metoda dreptunghiului;
- metoda trapezului.

Pentru ambele metode se presupune că:

$$f:[a,b] \longrightarrow R$$

este o funcţie integrabilă, iar:

$$W_n = (\, a=x_0 < x_1 < \ldots < x_n=b \,)$$

este o diviziune echidistantă a intervalului $[a,b]$.

Se defineşte:

$$D(f) = \frac{b-a}{2}\sum_{i=1}^{n} f(x_i)$$

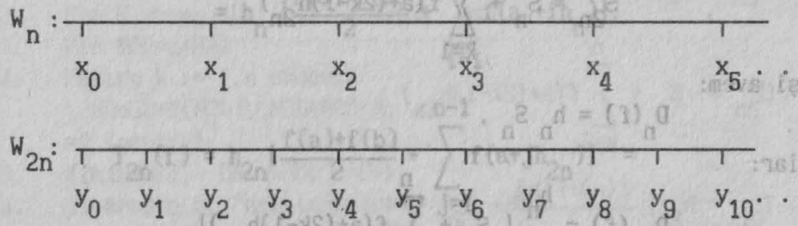
care aproximează f prin metoda dreptunghiului folosind n intervale. De asemenea,

$$T(f) = \frac{b-a}{2n}\left[f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_i)\right]$$

aproximează integrala I prin metoda trapezului folosind n intervale.

O modalitate de rafinare a diviziunilor.

Pentru a putea urmări convergenţa ambelor metode, le vom aplica în mod repetat, pentru n din ce în ce mai mare. După ce s-a calculat o aproximaţie cu un n dat, următoarea diviziune va conţine nodurile ultimei diviziuni împreună cu punctele din mijloacele intervalelor diviziunii vechi. Dacă W_n şi W_{2n} sînt cele două diviziuni în discuţie, ele apar schiţate ca mai jos:



Dacă $h_n = \frac{b-a}{n}$, atunci $x_i = a + ih_n$, $i = \overline{0,n}$

$$h_{2n} = \frac{b-a}{2n} = \frac{h_n}{2} \text{ și } y_j = a + jh_{2n} = a + j\frac{h_n}{2}, \; j=\overline{o,2n}$$

Se observă că: $x_0 = y_0 = a$, $x_n = y_{2n} = b$, $x_k = y_{2k}$, $k=\overline{1,n-1}$.

Aproximări succesive prin metoda dreptunghiului:

$$D_n(f) = \frac{b-a}{n}\sum_{i=1}^{n} f(x_i) = h_n \sum_{i=1}^{n} f(a + ih_n)$$

$$D_{2n}(f) = h_{2n}\sum_{i=1}^{n} f(a+jh_{2n}) =$$

$$= h_{2n}\left[\sum_{k=1}^{n} f(a+2kh_{2n}) + \sum_{k=1}^{n} f(a+(2k-1)h_{2n})\right] =$$

$$= h_{2n}\left[\sum_{k=1}^{n} f(a+kh_n) + \sum_{k=1}^{n} f(a+(2k-1)h_{2n})\right].$$

Dacă notăm: $S_n = \sum_{i=1}^{n} f(x_i)$ și $S_{2n} = \sum_{j=1}^{n} f(y_j)$ atunci:

$$S_{2n} = S_n + \sum_{k=1} f(a+(2k-1)h_{2n})$$

și avem:

$$D_n(f) = h_n S_n ,$$

iar:

$$D_{2n}(f) = \frac{h_n}{2} [S_n + \sum_{k=1}^{n} f(a+(2k-1)h_{2n})].$$

Începem aproximările succesive cu $n = 1$ și avem:

$h_1 = b-a$; $S_1 = f(b)$; $D_1 = h_1 S_1$;

Pentru un n oarecare, relațiile de recurență sînt:

$$h_{2n} = h_n/2 \qquad (5.3.1)$$

$$S_{2n} = S_n + \sum_{k=1}^{n} f(a+(2k-1)h_{2n}) \qquad (5.3.2)$$

$$D_{2n} = h_{2n} S_{2n} \qquad (5.3.3)$$

Procedeul se aplică în mod repetat, pînă cînd:

$$| D_{2n} - D_n | < E \qquad (5.3.4)$$

unde E are o valoare fixată anterior.

Aproximări succesive prin metoda trapezului.

$$T_n(f) = \frac{b - a}{2n} [f(a) + f(b) + 2 \sum_{i=1}^{n} f(x_i)] =$$

$$= h_n [\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n} f(a+ih_n)].$$

$$T_{2n}(f) = h_{2n}[\frac{f(a)+f(b)}{2} + \sum_{j=1}^{n-1} f(a+jh_{2n})] =$$

$$= h_{2n}[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n} f(a+2kh_{2n}) + \sum_{k=1}^{n} f(a+(2k-1)h_{2n})] =$$

$$= h_{2n}[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n} f(a+kh_n) + \sum_{k=1}^{n} f(a+(2k-1)h_{2n})].$$

Dacă notăm: $S_n = \sum_{i=1}^{n} f(x_i)$ şi $S_{2n} = \sum_{j=1}^{n-1} f(y_j)$ atunci

$$S_{2n} = S_n + \sum_{k=1}^{n} f(a+(2k-1)h_{2n})$$

şi avem:

$$T_n(f) = h_n [\frac{f(a)+f(b)}{2} + S_n],$$

$$T_{2n}(f) = \frac{h_n}{2} [\frac{f(a)+f(b)}{2} + S_n + \sum_{k=1}^{n} f(a+(2k-1)h_{2n})].$$

Începem şi aici aproximările succesive cu *n* = 1 şi avem:

$$h_1 = b-a ; \quad S_1 = 0 ; \quad T_1 = h_1 [\frac{f(a)+f(b)}{2} + S_1].$$

Pentru un *n* oarecare relaţiile de recurenţă sînt:

$$h_{2n} = h_n/2 \qquad (5.3.5)$$

$$S_{2n} = S_n + \sum_{k=1}^{n} f(a+(2k-1)h_{2n})$$

$$T_{2n} = h_{2n}[\,\frac{f(a)+f(b)}{2} + S_{2n}\,]. \qquad (5.3.6)$$

Procedeul se aplică în mod repetat, pînă cînd:

$$|\,T_{2n} - T_n| < E\,, \qquad (5.3.7)$$

unde E are o valoare fixată anterior.

Conventii de programare.

În timpul calculelor, în formulele de recurență (5.3.1), (5.3.2), (5.3.6), nu ne interesează decît ultimii termeni. Vom nota deci acești ultimi termeni prin:

H - pasul h din formula 5.3.1;

SD - suma S_{2n} la metoda dreptunghiului, formula 5.3.2;

ST - suma T_{2n} la metoda trapezului, formula 5.3.6.

Pentru a verifica condițiile (5.3.4) si (5.3.7), sînt necesari ultimii doi termeni ai șirurilor de aproximații. Aceștia vor fi notați prin *DV*, *DN*, *TV* și *TN*.

DV și *DN* sînt respectiv penultima (vechea) și ultima (noua) aproximantă prin metoda dreptunghiului iar:

TV și *TN* sînt respectiv penultima (vechea) și ultima (noua) aproximantă prin metoda trapezului.

La metoda trapezului valoarea $[f(a)+f(b)]/2$ o vom calcula o singură dată la început.

Variabila *NOD* conține de fiecare dată valoarea nodului curent.

Variabilele n și k au semnificațiile din formulele de mai sus.

Descrierea algoritmilor:

Algoritmul dreptunghi este:

```
1.  Date f, a,b,E;
2.  Fie H =b-a; SD=f(b); DV=H*SD;  n=1;
3.  Fie NOD=a+H/2;
4.  Pentru k := 1,n execută
      SD=SD+f(NOD); NOD=NOD+H;
    sf {pentru}
5.  Fie H=H/2;  DN=H*SD; n=2*n;
6.  Tipăreşte n; "dreptunghiuri ==>"; DN
7.  Dacă |DN-DV| < E atunci STOP
                        altfel mergi la pasul 8.
8.  Fie DV=DN; continuă cu pasul 3.
```

Algoritmul trapez este:

```
1.  Date  f,a,b,E;
2.  Fie H=b-a; ST=0; FAB=(f(a)+f(b))/2; TV=H*FAB; n=1;
3.  Fie NOD =a+H/2;
4.  Pentru k := 1,n-1 execută
      ST=ST+f(NOD); NOD=NOD+H;
    sf {pentru};
5.  Fie H=H/2; TN=H*(FAB+ST); n=2*n;
6.  Tipăreşte n; "trapezuri ==>"; DN
7.  Daca |TN-TV | < E atunci STOP
                        altfel mergi la pasul 8.
8.  Fie TV=TN; continuă cu pasul 3.
```

Prezentarea programelor.

Cele două metode sînt descrise în cadrul aceluiaşi program "*APROXINT*". În secvenţa de instrucţiuni dintre 100 si 410 este programată metoda dreptunghiului iar între 600 şi 910 metoda trapezului.

Astfel, lansarea cu **RUN** sau **RUN 100** înseamna apelarea metodei dreptunghiului, iar **RUN 600** înseamnă apelarea metodei trapezului.

Prezentăm în tabelul de mai jos grupurile de instrucţiuni care descriu metoda dreptunghiului.

Acelaşi tabel descrie şi corespondenţa dintre algoritmul trapez şi program. Pentru aceasta trebuie ca toate numerele de instrucţiuni să fie mărite cu 500, iar la ultima linie cuvîntul "*dreptunghiuri*" se va înlocui cu "*trapezuri*".

Instrucțiune sau grup de instrucțiuni	Descrierea sumară a secvenței
100	Mesajul de început
110-150	Pasul 1 din algoritm
160-190	Pasul 2 din algoritm
200	Pasul 3 din algoritm
210-240	Pasul 4 din algoritm
250-270	Pasul 5 din algoritm
280-290	Pasul 6 tipărirea la o iterație
300	Pasul 7 cu salt la tipărire rezultat
310-320	Pasul 8 din algoritm
330-410	Tipărește integrala cerută, valoarea ei aproximativă și numărul de dreptunghiuri folosite.

Programul BASIC

```
1   REM Programul 21: APROXINT
2   REM ************************
99  REM
100 PRINT "Calcul aprox. al integralelor;
         metoda dreptunghiurilor"
109 REM
110 INPUT "f(x)=?";f$
120 DEF FN f(x)=VAL f$
130 INPUT "a =?";a
140 INPUT "b =?";b
150 INPUT "epsilon =?";eps
160 LET H=b-a
170 LET SD=FN f(b)
180 LET DV=H*SD
190 LET n=1
200 LET NOD=a+H/2
210 FOR k=1 TO n
220    LET SD=SD+FN f(NOD)
230    LET NOD=NOD+H
240 NEXT k
```

```
249  REM
250  LET H=H/2
260  LET DN=H*SD
270  LET n=2*n
279  REM
280  PRINT n;" dreptunghiuri ==> "
290  PRINT "             ";DN
299  REM
300  IF ABS (DN-DV)<eps THEN GO TO 330
309  REM
310  LET DV=DN
320  GO TO 200
329  REM
330  PRINT "Integrala de la ";a
340  PRINT "             la ";b
350  PRINT "din    f(x)  =  ";f$
360  PRINT "cu precizia     ";eps
370  PRINT "este            ";DN
380  PRINT
390  PRINT "S-au folosit ";n
400  PRINT "dreptunghiuri"
410  STOP
600  PRINT "Calcul aprox. al integralelor;
          metoda trapezelor"
609  REM
610  INPUT "f(x)=?";f$
620  DEF FN f(x)=VAL f$
630  INPUT "a =?";a
640  INPUT "b =?";b
650  INPUT "epsilon =?";eps
659  REM
660  LET H=b-a
670  LET FAB=(FN f(a)+FN f(b))/2: LET ST=0
680  LET TV=H*FAB
690  LET n=1
699  REM
700  LET NOD=a+H/2
710  FOR k=1 TO n
720     LET ST=ST+FN f(NOD)
730     LET NOD=NOD+H
740  NEXT k
```

```
749  REM
750  LET H=H/2
760  LET TN=H*(FAB+ST)
770  LET n=2*n
779  REM
780  PRINT n;" trapeze ==> "
790  PRINT "               ";TN
799  REM
800  IF ABS (TN-TV)<eps THEN GO TO 830
809  REM
810  LET TV=TN
820  GO TO 700
829  REM
830  PRINT "Integrala de la ";a
840  PRINT "              la ";b
850  PRINT "din   f(x)  =  ";f$
860  PRINT "cu precizia    ";eps
870  PRINT "este           ";TN
880  PRINT
890  PRINT "S-au folosit ";n
900  PRINT "trapeze"
910  STOP
```

Exemple de utilizare.

```
RUN 600
Calcul aprox. al integralelor;
   metoda trapezului
f(x)=?"x↑2"
a=?0
b=?2
epsilon=?0.001
2 trapeze ==>
             3
4 trapeze ==>
             2.75
8 trapeze ==>
             2.6875
16 trapeze ==>
             2.671875
```

```
32 trapeze ==>
               2.6679688
64 trapeze ==>
               2.6669922
Integrala de la 0
             la 2
din f(x) =   x^2
cu precizia  .001
este         2.6669922

S-au folosit 64
trapeze.
```

Tabelul de mai jos dă cîteva elemente prin care se pun în evidență performanțele celor două metode.

a	b	E	f	metoda	iterații	intervale	valoare	durata (sec)
0	2	0.001	x^2	dreptunghi	12	4096	2.6676431	600
				trapez	6	64	2.6669922	10
	1		e^{x^2}	dreptunghi	10	1024	1.4634912	200
				trapez	6	64	1.4627623	15

5.4. Programul 22: APLICINT

Enunțul problemei: Acest program are drept scop calculul aproximativ al unor mărimi remarcabile, ale căror formule sînt exprimate cu ajutorul unor integrale. Avem în vedere cinci astfel de formule. În toate se consideră o funcție reală și continuă

$f : [a,b] \longrightarrow R$

și dreptele din plan de ecuație x = a, x = b, y = 0.

Enumerăm mai jos cele cinci formule împreună cu condițiile suplimentare (condiții de tip "suficient") în care aceste formu-

le sînt adevărate.

1. Aria domeniului delimitat de cele trei drepte şi de graficul funcţiei·

$$A(f) = \int_a^b f(x)\,dx$$

2. Volumul corpului obţinut prin rotaţia domeniului delimitat de cele trei drepte şi de graficul funcţiei în jurul dreptei *y=0*.

$$V(f) = PI \int_a^b (f(x))\uparrow 2\,dx$$

3. Lungimea arcului de curbă *y=f(x)* între punctele *(a,f(a))* şi *(b,f(b))*, dacă *f* este derivabilă şi *f'* este continuă:

$$l(f) = \int_a^b SQR(1+(f'(x))\uparrow 2)\,dx$$

4. Aria suprafeţei de rotaţie în jurul lui *y=0* a domeniului delimitat de cele trei drepte şi de graficul funcţiei, dacă *f* este derivabilă şi *f'* este continuă:

$$Arot(f) = 2\,PI \int_a^b f(x)\,.SQR(1 + (f'(x))\uparrow 2)\,dx$$

5. Coordonatele *Xg* şi *Yg* ale centrului de greutate ale unei plăci omogene, avînd conturul descris de cele trei drepte şi de graficul funcţiei:

$$X_g = \frac{\int_a^b x\,f(x)\,dx}{\int_a^b f(x)\,dx}$$

$$Yg = \frac{\frac{1}{2}\int_a^b (f(x))\uparrow 2\,dx}{\int_a^b f(x)\,dx}$$

Formulele sînt valabile numai dacă $f > 0$.

Functionarea programului.

Programul primeşte la intrare valorile *a*, *b* şi, după caz, expresia funcţiei *f* (cazurile 1, 2, 5), expresia derivatei *f'* (cazul 3), expresiile lui *f* şi *f'* (cazul 4). Pe ecran se afişează un meniu principal, după care utilizatorul este invitat să tasteze una dintre tastele *1, 2, 3, 4, 5, 0*, pentru a-şi selecta cazul dorit sau pentru a termina execuţia.

După alegere programul calculează integrala(sau integralele) care apar în cazul solicitat. Evaluarea aproximativă a integralelor se face n metoda trapezului, cu precizia *0.01*. Dacă se doreşte o altă precizie, se va modifica corespunzător instrucţiunea 4110.

În timpul evaluării integralei, la fiecare iteraţie, în colţul din stînga sus al ecranului se tipăreşte valoarea aproximativă a integralei.Algoritmul de calcul aproximativ este cel descris la programul **APROXINT**.

La sfîrşit, pe ecran apare rezultatul cerut. Imaginea de pe ecran rămîne atîta timp cît nu se apasă nici o tastă. În momentul unei apăsări ecranul se şterge şi apare din nou meniul principal.

Descrierea programului.

Etichetarea instrucţiunilor s-a făcut în aşa fel încît să evidenţieze, într-o oarecare măsură,părţile mari ale programului. Fiecare secvenţă a programului principal începe la un multiplu de 100, iar subprogramele încep la multiplu de 1000. Structura programului este dată în tabelul de mai jos.

Eticheta de început a secvenţei	Descrierea sumară a secvenţei
100	Afişează meniul principal şi cere alegerea variantei dorite. Variabila v primeşte valoarea tastată. Dacă nu se tastează una din cifrele permise, atunci meniul reapare.
500	Citeşte valorile a şi b. Dacă a>=b, atunci citirea se repetă.
1000	Calculează şi tipăreşte cele cerute în cazul 1
1100	Calculează şi tipăreşte cele cerute în cazul 2
1200	Calculează şi tipăreşte cele cerute în cazul 3
1300	Calculează şi tipăreşte cele cerute în cazul 4
1400	Calculează şi tipăreşte cele cerute în cazul 5 Dacă aria domeniului (care apare la numitor în formule) nu este strict pozitivă, se dă un mesaj de eroare.

Secvenţele de mai sus, începind cu 1000, apelează pentru calcule subprogramele care urmează.

2000	Subprogram de citire a expresiei şi de definire a funcţiei f.
3000	Subprogram de citire a expresiei şi de definire a derivatei f'.

Programul **APLICINT** nu verifică, în nici un fel, corectitudinea expresiilor pentru *f* şi *f'*, nici dacă *f'* este intr-adevăr derivata lui *f*.

4000	Subprogram de calcul aproximativ al unei integrale prin metoda trapezului. Secvenţa de calcul este aproape identică cu cea corespunzătoare de la adresa 600 din programul APROXINT. Pentru calculul valorii funcţiei de integrat într-un punct x, se apelează subprogramul de la 5000, care întoarce în variabila f această valoare. Subprogramul întoarce valoarea aproximativă în variabila TN.
5000	Subprogram care calculează valoarea funcţiei de integrat într-un punct x. Alegerea funcţiei de integrat o face folosind variabila v fixată în meniul principal şi care dă cazul solicitat.
6000	Subprogram care ajută la tipărirea rezultatelor finale. El tipăreşte valorile a, b, iar în funcţie de cazul solicitat (prin variabila v), tipăreşte expresia lui f(x) şi/sau a lui f'(x).
7000	Este o secvenţă de două instrucţiuni care aşteaptă, după afişarea rezultatelor, apăsarea unei taste. In momentul apăsării, pe ecran reapare meniul principal.

Programul BASIC

```
1   REM Programul 22: APLICINT
2   REM ************************
100 CLS: PRINT "Pentru o functie continua:"
110 PRINT "   f:[a,b]--->R": PRINT
120 PRINT "si pentru domeniul delimitat de:"
130 PRINT "  x=a, x=b, y=0, y=f(x)": PRINT
140 PRINT "va putem ajuta sa calculati:"
150 PRINT
160 PRINT "1. Aria domeniului"
170 PRINT "2. Volumul corpului de rotatie
          (in jurul lui y=0)"
180 PRINT "3. Lungimea graficului y=f(x)
          (daca f' continua)"
```

```
190  PRINT "4. Aria suprafetei laterale
           de rotatie (in jurul lui
           y=o, daca f' continua)"
200  PRINT "5. Coordonatele Xg si Yg ale centrului de greutate
           (daca f' > 0)"
210  PRINT
220  INPUT "Ce doriti?
          (1,2,3,4,5 sau 0 pentru STOP)";v
230  IF v=0 THEN STOP
499  REM
500  CLS
510  INPUT "a =?";a
520  INPUT "b =?";b
999  REM
1000 IF v<>1 THEN GO TO 1100
1010 GO SUB 2000
1020 GO SUB 4000
1030 GO SUB 6000
1040 PRINT "Aria domeniului =": PRINT "  ";TN
1050 GO TO 7000
1099 REM
1100 IF v<>2 THEN GO TO 1200
1110 GO SUB 2000
1120 GO SUB 4000
1130 GO SUB 6000
1140 PRINT "Volumul corpului =": PRINT "  ";PI*TN
1150 GO TO 7000
1199 REM
1200 IF v<>3 THEN GO TO 1300
1210 GO SUB 3000
1220 GO SUB 4000
1230 GO SUB 6000
1240 PRINT "Lungimea graficului =": PRINT "  ";TN
1250 GO TO 7000
1300 IF v<>4 THEN GO TO 1400
1310 GO SUB 2000
1320 GO SUB 3000
1330 GO SUB 4000
1340 GO SUB 6000
1350 PRINT "Aria laterala de rotatie =": PRINT "  ";2*PI*TN
1360 GO TO 7000
```

```
1399 REM
1400 IF v<>5 THEN GO TO 100
1410 GO SUB 2000
1420 LET v=1: GO SUB 4000
1430 IF TN<1e-7 THEN GO SUB 6000: PRINT "f este nepozitiva":
                         GO TO 7000
1440 LET aria=TN: LET v=2: GO SUB 4000
1450 LET Yg=TN/aria/2: LET v=5
1460 GO SUB 4000
1470 GO SUB 6000
1480 PRINT " Xg = ";TN/aria: PRINT " Yg = ";Yg
1490 GO TO 7000
1999 REM
2000 INPUT "f(x)=?";f$
2010 DEF FN f(x)=VAL f$
2020 RETURN
2999 REM
3000 INPUT "f'(x)=?";p$
3010 DEF FN p(x)=VAL p$
3020 RETURN
3999 REM
4000 LET H=b-a
4010 LET x=a: GO SUB 5000: LET FAB=f: LET x=b: GO SUB 5000:
     LET FAB=(FAB+f)/2
4020 LET ST=0: LET TV=H*FAB: LET n=1
1140 PRINT "Volumul corpului =": PRINT " ";PI*TN4030 LET NOD=a+H/2
4040 FOR k=1 TO n
4050    LET x=NOD: GO SUB 5000: LET ST=ST+f
4060    LET NOD=NOD+H
4070 NEXT k
4080 LET H=H/2
4090 LET TN=H*(FAB+ST)
4100 LET n=2*n
4110 LET TV=ABS (TN-TV): IF TV<0.01 THEN RETURN
4120 PRINT AT 0,0;TV: LET TV=TN
4130 GO TO 4030
4999 REM
5000 REM Alegerea functiei de sub integrala
5010 IF v=1 THEN LET f=FN f(x): RETURN
5020 IF v=2 THEN LET f=FN f(x)*FN f(x): RETURN
5030 IF v=3 THEN LET f=SQR (1+FN p(x)*FN p(x)): RETURN
```

```
5040 IF v=4 THEN LET f=FN f(x)*SQR (1+FN p(x)*FN p(x)): RETURN
5050 IF v=5 THEN LET f=x*FN f(x): RETURN
5999 REM
6000 CLS
6010 PRINT "Pentru:": PRINT
6020 PRINT " a = ";a
6030 PRINT " b = ";b
6040 IF v<3 OR v=5 THEN PRINT " f(x) = ";f$
6050 IF v=3 THEN PRINT " f'(x) = ";p$
6060 IF v=4 THEN PRINT " f(x)  = ";f$: PRINT " f'(x) = ";p$
6070 PRINT: PRINT "====>": PRINT
6080 RETURN
6999 REM
7000 IF INKEY$="" THEN GO TO 7000
7010 GO TO 100
```

Exemplu de utilizare:

[Pentru exemplificare, să considerăm $f : [0,1] \longrightarrow R, \quad f(x)=x$. Vom descrie mai întîi dialogul complet pentru cazul 1, deci pentru calculul ariei.]

```
RUN
Pentru o functie continua:
          f : [a,b] ---> R,
şi pentru domeniul delimitat de:
   x = a, x = b, y = 0, y = f(x)
va putem ajuta sa calculati:

1. Aria domeniului
2. Volumul corpului de rotatie
   (in jurul lui y=0)
3. Lungimea graficului y=f(x)
   (daca f' continua)
4. Aria suprafetei laterale de
   rotatie (in jurul lui y=0,
   daca f' continua)
5. Coordonatele Xg si Yg ale
   centrului de greutate
   (daca f > 0)
```

```
Ce doriti?
(1,2,3,4,5 sau 0 pentru STOP)1
a = ?0
b = ?1
f(x) = ?"x"
```

[Se afişează pe ecran, în stînga sus aproximaţiile integralei.]

```
Pentru:
a = 0
b = 1
f(x) = x
====>

Aria domeniului =
  0.5
```

[Rezultatele rămîn pe ecran pînă la apăsarea unei taste, după care reapare meniul principal. Dacă la cererea meniului principal se tastează 2, dialogul rămîne acelaşi, cu excepţia ultielor două rînduri de la rezultatele finale, care vor fi:]

```
Volumul corpului =
  1.0553788
```

[In mod analog, dacă se tastează 5, ultimele două rînduri vor fi:]

```
Xg = 0.671875
Yg = 0.3359375
```

[Dacă se tastează 3, în loc de f(x) =x va apare f'(x) = 1, iar ultimele două rînduri vor fi:]

```
Lungimea graficului =
  1.4142136
```

[In sfîrşit, dacă se tastează 4, va apare atît f(x)=x cît şi f'(x)=1, iar ultimele două rînduri vor fi:]

```
Aria laterala de rotatie =
  4.4428829.
```

5.5. Programul 24: OPALG.

Enunţul problemei. Să se efectueze operaţii cu matrice:
- suma a două matrice;
- produsul a două matrice;
- inversa unei matrice;
- calculul determinantului;
- rezolvarea unui sistem Cramer.

Descrierea programului. Programul a fost elaborat de elevii Bidian Sorin şi Frenţiu Alin de la Liceul G.Bariţiu şi prezentat de cei doi elevi la faza judeţeană a Sesiunii elevilor. Algoritmii folosiţi sînt cei din manualele de liceu. Intrucît deja am prezentat în 4.3 algoritmi pentru efectuarea operaţiilor de adunare şi înmulţire a două matrice nu insistăm aici asupra lor.

Programul este util nu numai pentru efectuarea operaţiilor menţionate mai sus, ci poate fi folosit şi pentru a examina textul Basic şi a urmări modul în care au fost realizate unele afişări de rezultate.

Programul se poate folosi uşor întrucît, prin text explicativ, cere utilizatorului informaţiile necesare.

Folosirea programului. La comanda RUN apare un text pe cîteva ecrane, prin care se arată scopul programului şi modul de utilizare. După acest text urmează:

```
IN FUNCTIE DE CE OPERATIE VRETI
SA EFECTUATI APASATI TASTA INDICATA

A - ADUNARE DE MATRICE
B - INMULTIRE DE MATRICE
R - INVERSA UNEI MATRICE
D - CALCULUL UNUI DETERMINANT
S - SISTEM DE ECUATII DE TIP
        CRAMER

DACA DORITI UN EXEMPLU TASTATI E
URMAT DE TASTA CORESPUNZATOARE
OPERATIEI DORITE
A
```

```
VA ROG INTRODUCETI DATELE

NUMAR DE LINII L = ?2
NUMAR DE COLOANE C =?2

A(1,1)=?10
A(1,2)=?20
A(2,1)=?30
A(2,2)=?40

B(1,1)=?100
B(1,2)=?200
B(2,1)=?300
B(2,2)=?400

TERMENII MATRICEI REZULTATE VOR
FI
C(1,1)=110
C(1,2)=220
C(2,1)=330
C(2,2)=440

PENTRU REVENIRE LA INCEPUTUL
PROGRAMULUI APASATI O TASTA
M
```

[Apare din nou "meniul", deci se așteaptă o nouă tastă]

```
D
VA ROG INTRODUCETI DATELE

NUMAR DE LINII SI COLOANE ?3
A(1,1)=?1
A(1,2)=?1
A(1,3)=?1
A(2,1)=?2
A(2,2)=?3
A(2,3)=?5
A(3,1)=?4
A(3,2)=?9
A(3,3)=?25
```

```
VALOAREA DETERMINANTULUI ESTE
  D=6

PENTRU A REVENI LA INCEPUTUL
PROGRAMULUI APASATI O TASTA

M
S
VA ROG INTRODUCETI DATELE

NUMAR DE ECUATII N=?3
A(1,1)=?1
A(1,2)=?1
A(1,3)=?1

A(2,1)=?2
A(2,2)=?4
A(2,3)=?6
A(3,1)=?350
A(3,2)=?5
A(3,3)=?0
TERMENUL LIBER B(1)=?6
TERMENUL LIBER B(2)=?28
TERMENUL LIBER B(3)=?360

NECUNOSCUTA N 1="X"
NECUNOSCUTA N 2="Y"
NECUNOSCUTA N 3="Z"

SOLUTIA SISTEMULUI ESTE

X=1
Y=2
Z=3
```

5.6. Programul 25: LOCGEO.

Introducere.

Una din cele mai dificile părţi ale geometriei sintetice este predarea noţiunii de loc geometric. Prin loc geometric se înţelege mulţimea punctelor din plan sau spaţiu care au o aceeaşi proprietate. În astfel de probleme apare un punct variabil, punct care aparţine unei mulţimi date de puncte (segment, cerc) sau verifică o relaţie algebrică. Pentru fiecare poziţie a punctului variabil se construieşte, după regulile date în problemă, punctul al cărui loc geometric se caută. Dificultatea principală a acestor probleme constă în a "ghici" proprietatea invariantă a punctului loc geometric atunci cînd punctul variabil parcurge mulţimea specificată. De regulă, în geometria sintetică plană locurile geometrice sînt compuse din drepte, cercuri, segmente sau arce de cerc. După ce elevul construieşte 2-3 figuri pentru 2-3 poziţii ale punctului variabil, el trebuie să intuiască "mişcarea" întregii figuri, în particular a locului geometric. Experienţa arată că acesta este momentul cel mai dificil în rezolvarea problemelor de loc geometric.

Lucrarea de faţă prezintă o modalitate prin care poate fi folosit calculatorul la "ghicirea" locurilor geometrice. Pentru aceasta s-a definit un limbaj, uşor accesibil atît profesorilor cît şi elevilor, prin care se poate descrie o clasă foarte largă de probleme de loc geometric. Avem în vedere, pînă în prezent, problemele în care punctul variabil aparţine unui segment sau unui cerc (eventual arc de cerc), iar în cadrul problemei apar segmente şi cercuri, puncte obţinute prin intersecţii sau care împart un segment într-un raport dat.

După descrierea problemei, calculatorul desenează îm mod repetat, pentru mai multe poziţii ale punctului variabil, figura ataşată problemei, păstrînd de la figurile precedente numai poziţiile punctului loc geometric. În acest mod "ghicirea" locului geometric nu mai este o problemă .

Folosind acest instrument, profesorul poate să dezvolte capacitatea de a intui locurile geometrice, făcînd demonstraţii cu calculatorul, pentru diverse tipuri de probleme.

Descrierea limbajului.

Limbajul folosit pentru descrierea problemelor de loc geometric permite:

- definirea punctelor;
- desenarea unor figuri ca: segment, dreaptă, perpendiculară, mediană, mediatoare, bisectoare, înălţime, cerc, tangentă etc.
- obţinerea unor puncte prin intersecţii.

Pentru definirea unui punct vom folosi propoziţia:

PCT *punct* = x , y

unde "*punct*" este numele punctului definit (şi este o majusculă a alfabetului latin), iar *x* şi *y* sînt coordonatele acestui punct. Ecranul televizorului este doar o parte a primului cadran, de aceea trebuie ca 0 <= x <= 255 şi 0 <= y <= 175.

Pe lîngă definirea explicită a unui punct, frecvent se spune că "*M este mijlocul segmentului AB*", sau că "*M imparte segmentul AB intr-un raport dat*". Se poate defini un punct pe dreapta AB şi prin poziţia sa faţă de punctele *A* şi *B* astfel:

M = (1-a) A + a B

cu a < 0	pentru	M la stînga lui A,
a = 0	pentru	M = A
0 < a < 1	pentru	M între A şi B
a = 1	pentru	M = B
a > 1	pentru	M la dreapta lui B

Pentru aceasta vom folosi propoziţia:

PCT M = A, B, *a*

unde "a" este fie o constantă reală, fie o variabilă. Dacă *h* şi *l* sînt două constante sau două variabile desemnate prin două litere mici, reprezentînd numere calculate anterior (vezi propoziţiile FIE şi DIST), atunci prin:

PCT M = A, B, *h/l*

se defineşte punctul *M*, care se află pe dreapta *AB* şi împarte segmentul în raportul *h/l*.

Pentru a obţine valoarea unei variabile oarecare *x* (*x* fiind o literă mică), se pot folosi propoziţiile :

FIE *x* = *numar*

sau:

DIST *x* = *A*, *B*

În prima propoziție se ia valoarea numărului precizat, iar în cazul al doilea se ia ca valoare lungimea segmentului [*AB*].

Pentru a trasa segmentul *AB* vom folosi propoziția:

SEGMENT *A* , *B*

iar propoziția:

DREAPTA *A*, *B*

va avea ca efect desenarea dreptei determinate de punctele *A* și *B*. Pentru a defini și trasa dreapta ce trece prin punctul *A* și este paralelă cu dreapta *BC* vom folosi propoziția:

DREAPTA *A*, (*B*, *C*)

Se acceptă și prescurtările **SEGM**, respectiv **DR** pentru **SEGMENT**, respectiv **DREAPTA**.

Propozițiile

CERC O, *r*

respectiv:

CERC *A*, *B*, *C*

au ca efect trasarea unui cerc, precizat fie prin centrul *O* și raza *r*, fie prin trei puncte *A*, *B*, *C*. În primul caz raza *r* trebuie să fie definită printr-una din propozițiile **FIE** sau **DIST**, prezentate mai sus, sau este o constantă.

Pentru a stabili centrul cercului descris în cazul al doilea se folosește propoziția:

CENTRU (*A*, *B*, *C*) = *O*

unde *O* este o literă mare care marchează centrul cercului.

În mod frecvent, apare necesitatea definirii unui punct pe un cerc. Pentru această situație presupunem că centrul cercului este deja dat printr-un punct, iar raza lui este dată printr-o

propoziţie FIE sau DIST. Pentru a preciza poziţia punctului pe cerc se mai cere unghiul (în radiani), în sens trigonometric, faţă de diametrul orizontal al cercului. Acest unghi se dă fie printr-o constantă, fie printr-o variabilă a cărei valoare a fost specificată anterior. Propoziţia prin care se defineşte acest punct este:

PCT *punct* = *centru*, *raza*, *unghi*

unde "*punct*" şi "*centru*" sînt puncte date prin cîte o literă mare, iar *raza* şi *unghi* sînt nume de variabile date prin cîte o literă mică, sau constante.

În sfîrşit, o altă posibilitate de a defini un punct este intersecţia unei tangente cu un cerc. Pentru un punct dat *A* exterior unui cerc există două tangente la cerc, deci două puncte de tangenţă, de exemplu *R* şi *T*. Pentru a defini aceste puncte se va folosi propoziţia:

PCT *R*, *T* = TANG *A* CERC *O*, *r*

sau:

PCT *R*, *T* = TANG *A* CERC *M*, *N*, *P*

Pentru a trasa una dintre tangente vom desena dreapta care trece prin punctele *A* şi *R* (sau *A* şi *T*).

În cazul cînd *A* se află pe cerc cele două puncte se consideră confundate şi se folosesc pentru a reţine direcţia tangentei. Astfel,

PCT *P*, *P* = TANG *A* CERC *O*, *r*

defineşte punctul *P*(spre marginea ecranului) astfel că dreapta *AP* este tangentă la cerc. Tangenta este desenată cu:

DR *A*, *P*

În enunţul problemelor de loc geometric intervine un element variabil (un punct variabil pe un segment dat, un punct variabil pe un cerc, etc). Pentru a defini un astfel de punct se foloseşte propoziţia:

VAR *punct* SEGMENT *stinga dreapta*

sau:

VAR *punct* CERC *centru raza*

în care "*punct*", "*stinga*", "*dreapta*" şi "*centru*" sînt date cu cîte o majusculă, iar raza cu o minusculă. In momentul definirii punctului variabil,se cer valoarea minimă şi cea maximă a parametrului de descriere a segmentului(o valoare din intervalul [0,1]) sau cercului (unghiul în radiani). De asemenea se cere numărul de puncte ale locului geometric care se doresc a fi afişate.

Prin propoziţia:

LOC *punct*

se defineşte locul geometric, ca fiind mulţimea punctelor "*punct*".

Prin propoziţia:

INT (*elem1*, *elem2*) = **PCT** *p1* [,*p2*]

se defineşte un punct (sau două) ca intersecţie a două figuri geometrice precizate prin *elem1* şi *elem2*. Acestea pot fi sau drepte sau cercuri. În cazul în care ambele sînt linii drepte rezultatul este un singur punct, iar în cazul intersecţiei unei drepte cu un cerc, sau a două cercuri, se definesc două puncte. (În descrierea propoziţiei parantezele drepte nu fac parte din sintaxă, reprezintă faptul că *p2* este opţional.)

Prin propoziţia:

PERP *B*, (*C*, *D*), *P*

se defineşte o dreapta ce trece prin punctul *B* şi este perpendiculară pe *CD*. *P* este punctul în care perpendiculara taie dreapta *CD*.

Prin propoziţia:

PERP *B*, *A*, *B*, *P*

se cere trasarea perpendicularei în *B* pe dreapta *AB*. Se defineşte şi un punct (spre marginea ecranului) astfel încît *BP* este perpendiculara dorită. Se observă că parantezele folosite mai sus sînt opţionale, ca de altfel şi celelalte semne de punctuaţie, care au doar rolul de a separa punctele. Astfel scrierea:

PERP *B A B P*

este corectă dar:

PERP *B* *AB* *P*

este greşită întrucît *A* şi *B* nu sînt separate de nici un caracter.

Prin propoziţia:

MEDN *A*, (*B*, *C*), *P*

se defineşte mediana *AP* dusă din *A* pe latura *BC* a triunghiului.

Prin propoziţia:

MEDT *A*, *B*

se defineşte mediatoarea segmentului AB.

Pentru definirea bisectoarei avem propoziţia:

BISECT *A*, *B*, *C*, *P*

care defineşte bisectoarea unghiului *ABC*, *P* fiind punctul în care bisectoarea taie latura opusă *AC*.

Înălţimea unui triunghi poate fi definită prin propoziţia:

INALT *A*, (*B*, *C*), *P*

care reprezintă înălţimea *AP* dusă din *A* pe *BC*.

Se pot defini şi triunghiuri prin propoziţia:

TRIUNGHI *A*, *B*, *C*

Exemplul 1. Fie *A* un punct fix şi *M* un punct mobil pe un cerc dat. Să se afle locul geometric al punctului *P*, mijlocul segmentului *AB*.

Descrierea acestei probleme în limbajul definit mai sus este:

```
FIE r = 40
PCT A = 10 , 20
PCT C = 130 , 90
CERC C , r
VAR M CERC C , r
PCT P = A , M , 0.5
LOC P
```

Exemplul 2. Fie *A* un punct fix și *M* un punct mobil pe un cerc dat. Să se afle locul geometric al mijlocului coardei determinate de dreapta *AM*.

Descrierea problemei este :

```
PCT A = 10 , 10
FIE r = 40
PCT O = 128 , 88
CERC O , r
VAR M CERC O , r
INT ( DREAPTA (A , M) ; CERC (O , r) ) = PCT M , Q
PCT P = Q , M , 0.5
LOC P
```

Exemplul 3. În triunghiul dreptunghic *ABC* ipotenuza *BC* este fixă, iar vîrful *A* este variabil. Se prelungește *BA* cu *AD* = *BA*, se unește mijlocul *E* al laturii *BC* cu *D* și se notează cu *M* punctul de intersecție dintre dreptele *ED* și *AC*. Se cere locul geometric al punctului *M*.

Descrierea problemei este următoarea:

```
PCT  B = 70 , 90
PCT  C =  150 , 90
PCT  E = B , C , 0.5
DIST  r = E , B
CERC  E, r
VAR  A  CERC  (E , r)
PCT  D =  B , A , 2
INT  ( DREAPTA  E , D  ,  DREAPTA  A , C) = PCT  M
LOC  M
```

Exemplul 4. Se dă un triunghi *ABM* înscris într-un cerc, cu vîrfurile *A* și *B* fixe, și *M* variabil. Se cere locul geometric al centrului de greutate al triunghiului.

Descrierea problemei:

```
PCT   A = 90 ,  40
PCT   B = 160 , 40
PCT   C = 125 ,  170
CERC  A , B  , C
CENTRU A , B, C  = O
```

```
DIST r = 0 , B
VAR  M  CERC O , r
TRIUNGHI A , B , M
PCT  R = A , M ,  0.5
PCT  D = A , B ,  0.5
INT ( DREAPTA  M , D , DREAPTA  B , R ) = PCT G
LOC G
```

Exemplul 5. Se dă un triunghi *ABC*, *AB*=*AC*. Pe [*AB*] se ia un punct *M*, iar pe [*AC*] un punct *N* astfel încît *AM*=*CN*. Se cere locul geometric al mijloacelor segmentelor *MN* cînd *M* parcurge segmentul *AB*.

Descrierea problemei este:

```
PCT A 125  170
PCT B 85 10
PCT C 165 10
TRIUNGHI A B C
VAR M SEGM  A B
DIST a  A B
DIST x  A M
PCT N  C A x/a
SEGM  M N
PCT L  M N 1/2
LOC L
```

Exemplul 6. Se dau două cercuri concentrice de raze *r* și *s* (*s*>*r*). Pe cercul de rază *r* se alege un punct fix *P*, iar pe cercul de rază *s* se alege un punct variabil *B*. Dreapta *BP* intersectează a doua oară cercul de rază *s* în *C*, iar perpendiculara în *P* pe dreapta *BC* intersectează cercul de rază *r* în *A* (*A* coincide cu *P* dacă această perpendiculară este tangentă la cercul de rază *r*). Se cere locul geometric al mijloacelor segmentelor *AB* cînd *B* parcurge cercul mare. (Problemă dată la Olimpiada internațională, Canbera, 1988).

Descrierea problemei:

```
PCT  O = 128 , 88
FIE  s =  76
```

```
FIE r = 55
CERC 0 r
CERC 0 s
PCT P 0 s 0.
VAR B CERC 0 s
INT DR B P CERC 0 r PCT P C
PERP P B P X
INT DR P X CERC 0 r PCT A P
PCT M A B 0.5
LOC M
```

Se poate remarca faptul că pentru o problemă există mai multe moduri de descriere care dau acelaşi rezultat. În limbajul descris pentru separator se poate folosi orice caracter diferit de litere, cifre şi punct. În exemplele de mai sus s-au folosit virgula, spaţiul şi parantezele.

Utilizarea programului.

Programul **LoGeo** este scris în **BASIC** pentru calculatoarele personale compatibile cu **Sinclair ZX Spectrum** (**HC-85**, **Tim-S**, **Cobra**, etc), şi poate fi obţinut direct de la autori.

După lansarea în execuţie cu comanda **RUN**, după un scurt semnal, programul cere introducerea problemei în limbajul definit în paragraful anterior. Fiecare instrucţiune (propoziţie) se introduce pe cîte o linie de cel mult *80* caractere. Sfîrşitul unei linii se marchează prin apăsarea tastei **ENTER** (sau **CR**). Ultima linie trebuie să conţină un singur caracter **$**. În acest moment pe ecran apare un mic meniu sub forma:

L **listare** *R* **rulare** *E* **editare**
I **inserare** *S* **oprire**

Dacă se tastează *L* (fără a mai tasta şi **ENTER**), pe ecran apare textul problemei, fiecare linie avînd un număr de ordine.

Dacă se tastează *R*, programul desenează pe ecran problema specificată, marcînd prin puncte locul geometric.

Erorile de sintaxă a propoziţiilor sînt verificate la prima execuţie a figurii. La întîlnirea primei erori, se dă numărul liniei eronate cu un mesaj de eroare şi se trece la meniu.

Dacă în timpul trasării unei figuri se apasă o tastă oareca-

re, atunci la terminarea desenării figurii imaginea se oprește. La o nouă apăsare a unei taste se construiește figura următoare.

Dacă în timpul desenării unei figuri programul depistează o eroare, cum ar fi:

- cerc dat prin trei puncte coliniare;
- raportul a două numere cu numitor nul;
- un punct iese în afara ecranului de [0,255] x [0,175];
- tangenta la cerc dintr-un punct interior;
- intersecția a două elemente este mulțimea vidă;
- cerc de rază negativă, etc,

atunci se dă un mesaj de eroare. Dacă fenomenul apare la prima figură, atunci după mesajul de eroare se revine la meniu, dacă apare la figurile următoare, atunci se trece la o nouă figură pentru poziția următoare a punctului variabil.

Dacă se tastează *E* în meniu, programul intră în mod de editare, permițînd modificarea oricărei linii. Se cere numărul liniei de modificat (de editat). Pe ecran apare linia specificată cu marcarea pozițiilor din linie. Se poate specifica o secvență din această linie care urmează a fi modificată. Specificarea se face prin pozițiile de început și de sfîrșit, care pot fi și identice, în care caz se modifică o singură poziție. Noua secvență care va înlocui pe cea specificată se introduce după ce apare în linia de jos a ecranului simbolul "*". Cele două secvențe pot avea și lungimi diferite. Dacă noua secvență este vidă (după * se introduce imediat **ENTER**) secvența veche se șterge. Dacă se șterg toate caracterele unei linii, se șterge întreaga linie.

Inserarea unei secvențe după o poziție dată se face prin înlocuirea caracterului respecttiv cu o secvență care începe cu același caracter. De exemplu, în linia:

```
         1         2         3
123456789012345678901234567890

     PCT A=90
```

vrem să inserăm *80* înaintea lui *90*. Procedăm astfel:

```
De la: 15 la: 15

* 80 , 9
```

Cea mai mare poziţie care poate fi specificată într-o editare este poziţia ultimului caracter diferit de spaţiu al liniei respective. După o editare apare din nou textul "*De la: la:*" aşteptînd o nouă editare a aceleiaşi linii. Editarea unei linii se termină dacă introducem *0* (zero) ca poziţie de început de secvenţă. (Poziţia de sfîrşit poate fi orice număr în acest caz.)

Inserarea unei linii noi se face cu comanda *I*. Se cere numărul liniei de inserat. Acest număr poate fi între *1* şi *n+1*, unde *n* este numărul ultimei linii. Inserarea se face înaintea liniei specificate.

Cu comanda *S* se termină execuţia programului. Dacă dorim să salvăm programul pe casetă împreună cu problema introdusă, acest lucru este posibil cu ajutorul comenzii **SAVE**. La o nouă încărcare, dacă se lansează execuţia nu cu **RUN**, ci cu **GO TO nr** (unde "*nr*" este numărul liniei unde începe meniul) programul începe execuţia cu afişarea meniului. Este bine să verificăm existenţa problemei păstrate prin lansarea comenzii *L*. Dacă dorim să salvăm programul fără problema dată, înainte de comanda **SAVE** se lansează comanda **CLEAR**.

O primă versiune a programului **LoGeo** este deja operaţională. Limbajul acceptat va fi extins după ce prezentul limbaj va fi suficient de mult experimentat, pentru a depista elemente noi necesare spre a-l face cît mai comod de utilizat.

5.7. Programul 25: ECINEL.

După o experienţă de rezolvitor şi de profesor la clasă se constată că rezolvarea anumitor tipuri de probleme se poate face cu ajutorul unor algoritmi. Programul pe care îl prezentăm are drept scop asigurarea unei asistenţe rezolvitorului (elevului) de probleme exponenţiale sau logaritmice. Asistenţa calculatorului constă în recomandarea unor algoritmi cu ajutorul cărora să se poată rezolva aceste probleme. Deci se doreşte a realiza o învăţare asistată de calculator a algoritmilor propuşi a fi utilizaţi în rezolvarea de ecuaţii sau inecuaţii exponenţiale sau logaritmice. Aşadar programul nu realizează o suplinire a informaţiilor din manualele şcolare, deoarece algoritmii prezentaţi nu se găsesc în ele, ci mai degrabă o sistematizare, o completare a cunoştinţelor dobîndite din manuale şi de la clasă. Aşa cum se de-

deduce, fiind un instrument util în învăţare, programul (algoritmii prezentaţi) este util elevilor care vor să înveţe cum se poate aborda rezolvarea problemelor de la capitolul ecuaţii şi inecuaţii exponenţiale sau logaritmice. Acest program se poate utiliza la clasă, dar mai util este în pregătirea individuală a elevilor. La clasă algoritmii din program pot fi prezentaţi elevilor şi sub forma unor planşe, fişe sau pe folie de retroproiector. Forma de prezentare a lor se alege în funcţie de dotarea materială şi este la dispoziţia profesorului.

Menţionăm că stabilirea algoritmilor şi realizarea programului s-au făcut cu participarea profesoarei Maria Toadere de la Liceul industrial nr. 8 din Cluj-Napoca.

In funcţie de problema concretă pe care utilizatorul programului o are de rezolvat, beneficiind de indicaţiile programului, el va efectua calculele singur. Elevul (utilizatorul) va furniza programului unele informaţii despre problema pe care o are de rezolvat, în funcţie de care se vor afişa indicaţii de lucru pentru rezolvarea problemei. Aşadar rezolvitorul îşi alege traseul corespunzător problemei pe care o are de rezolvat. Programul este conceput ca să dea indicaţii pentru oricîte probleme, pe rînd. După parcurgerea oricărui traseu programul revine la început şi deci, dacă rezolvitorul a ales un traseu greşit, el poate consulta şi alte trasee.

Algoritmii prezentaţi urmăresc transformarea ecuaţiilor sau a inecuaţiilor în una dintre formele particulare, precizate de către program, forme numite şi "puncte de trecere" deoarece după efectuarea transformării ecuaţia (inecuaţia) îşi schimbă forma exponenţială sau logaritmică într-o altă formă, polinomială de obicei, din care se vor afla soluţiile. Trebuie să precizăm că pentru inecuaţii se propun aceleaşi transformări ca şi pentru ecuaţii.

Deşi punctele de trecere sînt scopul final al transformărilor propuse, ele sînt plasate la începutul fiecărui traseu, de la ele făcîndu-se ramificări în funcţie de formele posibile ale ecuţiilor şi inecuaţiilor.

<u>Mod de utilizare</u>. După lansarea programului sau după sfîrşitul fiecărui traseu pe ecran apare meniul:

Programul ofera indicatii pentru rezolvarea de:

1 - ecuatii exponentiale

2 - ecuatii logaritmice

3 - inecuatii expon. sau log.

Tastati numarul corespunzator tipului de problema ce doriti sa o rezolvati sau 0 pt. stop

În locul prezentării funcţionării programului vom arăta pe rînd succesiunea conţinutului ecranelor de monitor pentru fiecare traseu în parte. Pentru a arăta că anumite informaţii apar în acelaşi ecran aceste informaţii vor fi încadrate în dreptunghiuri. Pentru fiecare ecran vom scrie în faţa sa, stînga sus, informaţia care a trebuit introdusă succesiv de la ultima apariţie a meniului pentru ca monitorul să conţină informaţiile respective. Informaţiile care trebuiesc introduse prin apăsarea tastei CR (RETURN) vor fi subliniate, iar dacă într-un anume loc pentru continuare se poate apăsa pe orice tastă (fără a fi necesară apăsarea tastei CR) acest lucru se precizează prin scrierea literei "c" (de la caracter)

1

Dorim sa ajungem la una dintre formele numite puncte de trecere

1. $a^{f(x)} = a^{g(x)}$

2. $a^{f(x)} = b$

3. $a^{f(x)} = b^{f(x)}$

4. $a^{f(x)} = b^{g(x}$

Dacă ecuaţia are una dintre forme tastaţi numărul respectiv altfel 0

11

Se va rezolva ecuatia:

$f(x)=g(x)$,

daca baza contine necunoscut si ecuatiile:

a=1 a=0 a=-1.

Solutiile se vor verifica.

Pentru a continua tastati orice

12

> Se va rezolva ecuatia:
>
> $f(x)=\log_a b$
>
> Iar daca exponentul(adica f(x)) este exprimat cu logaritmi in baza c se va rezolva ecuatia:
>
> $f(x)\log_c a = \log_c b$
>
> Pentru a continua tastati orice

13

> Se va rezolva ecuatia:
>
> $f(x)=0$
>
> daca o baza contine necunoscuta se vor rezolva si ecuatiile
>
> $a=b \quad a=-b.$
>
> Solutiile se vor verifica
>
> Pt.a continua tastati orice

14

> Se va rezolva ecuatia:
>
> $f(x)=g(x)\log_a b$
>
> Iar daca un exponent (f(x) sau g(x)) este exprimat cu logaritmi in baza c se va rezolva ecuatia:
>
> $f(x)\log_c a=g(x)\log_c b$
>
> Pentru a continua tastati orice

Dacă ecuaţia nu are forma unui punct de trecere mai întîi se vor face transformări în ce priveşte numărul de baze şi numărul de termeni ai ecuaţiei, iar apoi in funcţie de forma ecuaţiei se va preciza calea de urmat pentru a ajunge la un punct de trecere

sau, dacă ecuaţia nu are nici una dintre formele prestabilite, se vor indica şi unele cazuri exceptate împreună cu modul de abordare al acestora.

10

Se va considera baza numai daca puterea contine necunoscuta la exponent

Pentru a obtine mai putine baze aplicati formulele:

descomp. in factori(ex: $24=2^3 3$)

transf. fractiilor($0,25=1/4=2^{-2}$)

$a^x b^x = (ab)^x$

$a^{\log_b c} = c^{\log_b a}$

Pentru a continua tastati orice

10c

Se vor face grupari de termeni si se vor scoate factori comuni. Daca nu se poate ajunge la forma produs=0 atunci trebuie ca in paranteze sa nu ramina necunoscuta la exponent.

Atentie la formule, deoarece

$a^{3x} + a^{2x} = a^x(a^3 + a^2)$ E GRESIT

$a^{3x} + a^{2x} = a^x(a^{2x} + a^x)$ E CORECT

Pt.a continua tastati orice

10cc

Dati nr. de termeni ai ecuatiei

10cc2

Ecuatia are doi termeni
Dati numarul bazelor:

10cc21

Ecuatia este de forma:

$$pa^{f(x)} = qa^{g(x)}$$

Se inmulteste cu $a^{-g(x)}/p$ si se va ajunge la un punct de trecere

Pentru a continua tastati orice

10cc3

Ecuatia are trei termeni.
Dati numarul bazelor:

10cc22

Ecuatia poate fi:

1. $pa^{f(x)} = qb^{f(x)}$

dupa impartire cu $pb^{f(x)}$ se ajunge la un punct de trecere,
sau

2. $pa^{f(x)} = qb^{g(x)}$

si in acest caz
fie transformam exponentii pt. a deveni egali (forma 1),
fie se descompun coef. p si q in factori primi iar apoi se separa bazele.
Pt. a continua tastati orice

10cc31

Ecuatia are una dintre formele:

1. $pa^{2f(x)} + qa^{f(x)} + r = 0$

si notam $a^{f(x)} = y$, $y > 0$.

2. $pa^{f(x)} + qa^{-f(x)} + r = 0$

inmultind cu $a^{f(x)}$ se ajunge la forma 1.

3. $pa^{2f(x)} + qa^{f(x)+g(x)} + ra^{2g(x)} = 0$

inmultind cu $a^{-2g(x)}$ se ajunge la forma 1.
Pentru a continua tastati orice

10cc32

Ecuatia are una dintre formele

1. $pa^{2f(x)} + q(ab)^{f(x)} + rb^{2f(x)} = 0$

impartim cu $b^{2f(x)}$ si ajungem la o ecuatie de gradul 2

2. $pa^{f(x)} + qb^{f(x)} + r = 0$ dar

produsul bazelor=1(ex:a=3-sqr8 b=3 + sqr(8)), atunci

inmultind cu $a^{f(x)}$ se va

la o ecuatie de gradul 2.
Pt.a continua tastati orice

În cazul în care ecuația are mai mult de trei termeni sau are trei termeni și trei baze (după ce s-au făcut transformările propuse) atunci ecuația se consideră a fi de un tip exceptat și se propun două forme de asemenea ecuații urmînd ca rezolvitorul să decidă în ce caz este problema pe care dorește el să o rezolve. În acest caz conținutul ecranului este:

```
Situatii exceptate posibile:

1. Exista o valoare c astfel ca:
   membrul 1>=c si membrul 2<=c

atunci se rezolva sistemul:
fiecare membru = c.

        f(x)
ex: a        = cos g(x), cu f(x)>=0
si a>=1 sau f(x)<=0 si 0<a<=1
Deci c=1.

2. Se va observa ca solutie un
numar a, apoi pentru cazurile:
I. x < a  si  II. x > a se va
demonstra pe baza monotoniei ca
membrul I > (sau <) membrul II.

Pentru a continua tastati orice
```

Astfel au fost prezentate toate ramurile pentru ecuațiile exponențiale. Urmează acum să prezentăm traseele oferite de proram în cazul în care la apariția meniului se introduce 2.

2

Dorim sa ajungem la una dintre formele numite puncte de trecere

1. $\log_a f(x) = \log_a g(x)$

2. $\log_a f(x) = b$

Daca ecuatia are una dintre forme tastati numarul respectiv altfel 0

21

Se va rezolva ecuatia:

$f(x) = g(x)$

Apoi solutiile se:
-verifica in ec.initiala
sau
-verifica in conditiile puse
sau
-intersecteaza cu dom.de def

Pt. a continua tastati orice

22

Se va rezolva ecuatia:

$f(x)= a^b$

Apoi solutiile se:
- verifica in ec.initiala
sau
-verifica in conditiile puse
sau
-intersecteaza cu dom.de def.

Pentru a continua tastati orice

20

Se vor face transformarile:

1.Obtinerea unei baze unice cu formulele:

a) $\frac{\log_a b}{\log_a c} = \log_c b$

b) $\log_a b \log_c a = \log_c b$

c) $\log_a b = 1 / \log_b a$

d) $\log_{a^x} b = (1/x)\log b$

Pt.a continua tastati orice

20c

Toti logaritmii au aceeasi baza dar ecuatia:

1. Contine (dupa eventuala eliminare a numitorilor) produse sau puteri de logaritmi sub radicali
2. Nu contine nici o situatie dintre cele de la 1.

Obs. Se va considera logaritm numai daca exista necunoscuta in baza sa sau in expresia sa.

Tastati numarul corespunzator cazului ecuatiei.

20c1

Se vor face transformari utilizind formulele:

a) $\log_a A^b = b \log_a A$

b) $\log_a (AB) = \log_a A + \log_a B$

c) $\log_a (A/B) = \log_a A - \log_a B$

pentru ca in ecuație sa fie un singur logaritm. Se face substitutie si se va obtine o ecuatie algebrica.

Pt.a vedea punctele de trecere tastati 1 altfel tastati 2

20c2

Se va ajunge la punct de trecere daca se aplica formulele:

a) $b \log_a A = \log_a A^b$

b) $b = \log_a a^b$

c) $\log_a A + \log_a B = \log_a (AB)$

d) $\log_a A - \log_a B = \log_a (A/B)$

Pentru a vedea punctele de trecere tastati 1 altfel tastati 0

3

Daca inecuatia nu are forma unui punct de trecere se aplica transformarile date pentru ecuatii pina se va obtine un punct de trecere de tipul celor de la ecuatii.

Tastati
- Pentru transformarile:
  - de la exponentiale - 1,
  - de la logaritmice - 2,
- Pentru puncte de trecere:
  - de la exponentiale - 3,
  - de la logaritmice - 4,
- Pentru indicatii privind trecerea - 5,
- Pentru alta problema - 6,
- Pentru stop - 7.

35

Se va afla dom. de def., se va afla sol. inec. obtinute dupa trecere si cele doua multimi se vor intersecta

Pt. efectuarea trecerii in functie de baza deosebim cazurile:

1. baza are pozitie cunoscuta fata de 1

2. baza depinde de un parametru (alta litera decit necunoscuta)

3. baza depinde de necunoscuta.

Introduceti cazul punctului de trecere

351

Pentru baza > 1 se pastreaza sensul.

Pentru baza subunitara (intre 0 si 1) se schimba sensul.

Obs.E vorba despre sensul dintre expresiile la care se ajunge prin punctele de trecere de la ecuatii

352

Se face discutie alegind cazuririle:

1. baza > 1 caz in care se trece cu acelasi sens,

2. baza > 0 si baza < 1 caz in care se trece cu sens schimbat

Obs.E vorba despre sensul dintre expresiile la care se ajuge prin punctele de trecere de la ecuatii

353

Se rezolva sistemele de inec.:

I. baza > 1 si inecuatia obtinuta din punctul de trecere prin pastrarea sensului.

II. baza > 0 si baza < 1 si inecuatia obtinuta din punctul de trecere prin schimbarea sensului.

Pt.a continua tastati orice

6. PROGRAME PENTRU IBM-PC.

Limbajele GWBASIC,BASICA și TURBO BASIC sînt variantele limbajului BASIC utilizabile pe calculatoare IBM PC sau compatibile cu ele. În acest capitol prezentăm programele din capitolele precedente cu modificările minime necesare pentru a putea fi rulate pe calculatoare IBM PC. Înainte de textele sursă trecem în revistă instrucțiunile grafice, care diferă mult de cele utilizate în programele prezentate.

În mod grafic pot fi utilizate MaxX pixeli pe orizontală (axa OX), MaxY pixeli pe verticală (axa Oy), originea (0,0) fiind în colțul stînga sus, deci pixelii pe orizontală au adresele de la 0 la MaxX-1, iar pe verticală au adresele de la 0 la MaxY-1. Prezentăm cîteva valori posibile pentru MaxX și MaxY:

Ecran	MaxX	MaxY
EGA, rezoluție medie	320	200
EGA, rezoluție mare	640	200
EGA, rezoluție foarte mare	640	350
EGA, monocrom	640	350
VGA, rezolutie mare	640	480
Hercules(monocolor)	720	348

==

SCREEN - fixează modul de lucru al ecranului

==

Sintaxa : SCREEN [mod] [,[semnal-culoare]] [,[paga]] [,[pagv]]

mod = 0 - mod text, fără modificare lungime linie
1 - mod grafic, rezoluție medie, lățime linie 40
2 - mod grafic, rezoluție mare, lățime linie 80 (Obligatorie pentru ecran Hercules)
7 - mod grafic, rezoluție medie, EGA color
8 - mod grafic, rezoluție mare, EGA color
10 - mod grafic, rezoluție mare, EGA monocrom

11 - mod grafic, rezoluţie mare, VGA sau MCGA, alb şi negru.
12 - mod grafic, rezoluţie mare, VGA color

semnal-culoare -Are valoarea 0 sau 1, şi controlează informaţiile de culoare.
În mod text 0 : fără culoare
În rezoluţie medie 0 : cu culoare.

paga - întreg între 0 şi 7, reprezentînd pagina activă în care se scrie

pagv - întreg între 0 şi 7 reprezentînd pagina vizuală care apare pe ecran

---

PSET şi PRSET - desenează un punct pe ecranul grafic

Sintaxa:
```
PSET (x,y) [,culoare]
PRSET (x,y) [,culoare]
```

Desenează punctul de coordonate (x,y) cu culoarea specificată. Dacă culoarea este specificată, cele două forme acţionează la fel. Diferenţa între ele este că valoarea implicită a culorii la PSET este valoarea maximă posibilă, iar la PRSET această valoare este 0, ceea ce înseamnă că fără valoarea culorii PSET desénează, iar PRSET şterge punctul.

CIRCLE - desenează un cerc

Sintaxa :
```
CIRCLE (x,y), raza [,culoare [,început,sfîrşit [,aspect]]]
```

Desenează un cerc de centru (x,y), raza şi culoarea dată. Dacă se specifică începutul şi sfîrşitul în radiani atunci se desenează un arc de cerc. Pentru un cerc corect trebuie specificată valoarea variabilei aspect. Valoarea implicită este 5/6 pentru rezoluţie medie şi 5/12 pentru cea mare.

LINE - desenează un segment de dreaptă

Sintaxa:

LINE [(x1,y1)] - (x2,y2) [,[culoare] [,B[F]]

Desenează un segment ce unește punctul (x1,y1) cu (x2,y2), sau punctul curent cu (x2,y2) cu culoarea specificată. Cu obţiunea B desenează un dreptunghi, cu F îl colorează.

```
1   REM Programul 1 : EUCLID
2   REM ************************************************************
3   REM Algoritmul lui Euclid
4   REM Datele problemei  : n1, n2 - numere intregi
5   REM Rezultatul obtinut: d=(n1,n2), c.m.m.d.c al lui n1 si n2
6   REM ************************************************************
10  INPUT "n1="; N1
20  INPUT "n2="; N2
30  LET D=N1 : LET I=N2
40  IF I=0 THEN GOTO 100
50     LET Q=INT(D/I)
60     LET R=D-Q*I
70     LET D=I
80     LET I=R
90  GOTO 40
100 PRINT: PRINT "Cel mai mare divizor comun al numerelor"
110 PRINT "n1 ="; N1; ",  n2 ="; N2; "  este d ="; D
120 PRINT
130 STOP
```

```
1  REM Programul 2 : PROGRESIE
2  REM ***********************************************************
3  REM Progresie aritmetica cu ratia  r  si primul termen  a.
4  REM Se tiparesc primii  n  termeni
5  REM ***********************************************************
10 INPUT "Ratia ="; R
20 INPUT "Primul termen ="; A
30 INPUT "Numarul termenilor ="; N
40 LET T=A
50 FOR I=1 TO N
60   PRINT "t ("; I; ")="; T
70   LET T=T+R
80 NEXT I
90 STOP
```

```
1   REM Programul 3 : HORNER
2   REM ***********************************************************
3   REM Schema  lui  Horner
4   REM ***********************************************************
10  INPUT "Gradul polinomului= "; N
20  LET N1=N+1  30  DIM P(N1),Q(N1)
40  FOR I=1 TO N1
50      PRINT "P ("; I-1;") =";
60      INPUT P(I)
70  NEXT I
80  INPUT "a ="; A
90  LET Q(1)=P(1)
100 FOR I=2 TO N1
110     LET Q(I)=A*Q(I-1)+P(I)
120 NEXT I : PRINT
130 PRINT "Coeficientii citului din impartirea: P(X) / X-a sint"
140 PRINT
150 FOR I=1 TO N
160     PRINT "Q ("; I-1; ") ="; Q(I)
170 NEXT I
180 PRINT
190 PRINT "Restul impartirii ="; Q(N1)
200 PRINT
210 STOP
```

```
1   REM PROGRAMUL 4 : POLINOM
2   REM ********************************************************
3   REM Valoarea a doua polinoame P(X) si Q(X) pentru un X dat
4   REM ********************************************************
10  DEF FN F(T) = EXP(-T/2)*T-SIN(T)
20  INPUT "Gradul polinomului=";N
30  INPUT "Valoarea lui x="; X
40  LET N1=N+1
50  DIM P(N1)
60  FOR I=1 TO N1
70      LET P(I) = FN F(2*I+1)
80  NEXT I
90  GOSUB 200
100 PRINT "P("; X; ")="; V
110 REM Al doilea polinom este notat tot prin P intrucit asa
111 REM                  se cere in subprogram
120 FOR I=1 TO N1
130     LET P(I)=FN F(I*I+I+1)
140 NEXT I
150 GOSUB 200
160 PRINT "R("; X; ")="; V
170 STOP
200 REM  Subprogram care calculeaza  V = P(X)
201 REM      bazat pe schema lui Horner
210 LET V=P(1)
70  NEXT I220 FOR J=1 TO N1
230     LET V=V*X+P(J)
240 NEXT J
250 RETURN
```

```
1   REM programul 5 : CERC
2   REM ******************************************************
10  A=125: B=100: R1=75: R=20
20  INPUT "n=";N: INPUT "v=";V
30  K=1: SCREEN 1,0
40  FOR I=1 TO N
50   COLOR K: CLS
60   FOR AL=0 TO 6 283 STEP V
70     X=A+R1*COS(AL): Y=B-R1*SIN(AL)
```

```
80   IF AL <> 0 THEN CIRCLE (XO,YO),R,0
90   CIRCLE (X,Y),R,1
100  XO=X : YO=Y
110  NEXT AL
120  K=1+K
130 NEXT I
140 COLOR 0
```

```
1 REM Programul 6 : PATRAT
2 REM ************************************************************
10 DIM A(5): DIM B(5): DIM X(5): DIM Y(5)
20 FOR I=1 TO 4
30 PRINT "Coord.punctului nr.";I
40 INPUT "A=";A(I): INPUT "B=";B(I)
50 NEXT I
60 INPUT "Pas modif.latura ";PL
70 INPUT "Pas unghi(in grade) ";PU
80 LET PU=3.1415*PU/180
81 REM PU va fi in radiani
90 INPUT "Nr.de patrate ";N
100 INPUT "Nr.de cicluri ";M
110 LET L=SQR((A(2)-A(1))*(A(2)-A(1))+(B(2)-B(1))*(B(2)-B(1)))
120 LET A(5)=A(1): LET B(5)=B(1)
130 LET E=0: LET C=4
140 LET U=125: LET V=87
150 FOR I=1 TO M
160  SCREEN 1,0: CLS
170  FOR K=1 TO N
180    LET L1=L+(K-1)*PL: LET AL=(K-1)*PU
190    FOR J=1 TO 4
200      LET X(J)=INT(L1/L*(A(J)*COS(AL)-B(J)*SIN(AL))+.5)+U
210      LET Y(J)=INT(L1/L*(A(J)*SIN(AL)+B(J)*COS(AL))+.5)+V
220    NEXT J
230    LET X(5)=X(1): LET Y(5)=Y(1)
250    FOR J=2 TO 5
260     LINE (X(J-1),199-Y(J-1))-(X(J),199-Y(J))
270    NEXT J
280  NEXT K
```

```
290   LET E=E+1: IF E>7 THEN LET E=0
300   LET C=C+1: IF C>7 THEN LET C=0
310 NEXT I
320 STOP
```

```
1   REM Programul 7 : ECUATIE2
2   REM ************************************************************
3   REM Rezolvarea ecuatiei de grad II  ax^2 +bx + c = 0
4   REM ************************************************************
10  PRINT "Coeficientii ecuatiei:"
20  INPUT "a ="; A
30  INPUT "b ="; B
40  INPUT "c ="; C
50  CLS: PRINT
60  PRINT "Coeficientii ecuatiei sint:"
70  PRINT "a ="; A
80  PRINT "b ="; B
90  PRINT "c ="; C
100 IF A=0 THEN GOTO 300
110 LET D=B*B-4*A*C
120 PRINT "Valoarea discriminantului este  D ="; D
130 IF D<0 THEN GOTO 240
140 IF D=0 THEN GOTO 210
150 PRINT : PRINT "D>0, radacini reale distincte"
160 LET X1=(-B-SQR(D))/(2*A)
170 LET X2=(-B+SQR(D))/(2*A)
180 PRINT : PRINT "x1 ="; X1
190 PRINT "x2 ="; X2
200 GOTO 310
210 PRINT : PRINT "D=0, radacini reale confundate"
220 LET X1=-B/(2*A): LET X2=X1
230 GOTO 170
240 PRINT : PRINT "D<0, radacini complexe conjugate"
250 LET X1=-B/(2*A)
260 LET X2=SQR(-D)/(2*A)
270 PRINT : PRINT "Partea reala ="; X1
280 PRINT "Partea imaginara ="; X2
290 GOTO 310
```

```
300 PRINT : PRINT "Ecuatia nu este de gradul doi"
310 PRINT : PRINT "Dati alte valori (1=DA, 0=NU) ? ": INPUT N
320 IF N=1 THEN CLS: GOTO 10
330 STOP
```

```
1   REM Programul 8a : COMBI1
2   REM ************************************************************
3   REM Analiza combinatorie
4   REM ************************************************************
10  PRINT : PRINT "Datele de intrare"
20  INPUT "n ="; N : INPUT "k ="; K
30  CLS : PRINT "n ="; N: PRINT "k ="; K
40  IF N>33 THEN PRINT :PRINT "P ("; N
50  IF N<0 OR K<0 OR N<>INT(N) OR K<>INT(K) THEN GOTO 280
60  IF N=0 THEN PRINT: PRINT "Functiile A,C,P nedefinite":GOTO 290
70  REM  n! ----------------------------------------------------------
80  LET R=N
90  GOSUB 380
100 LET A2=F
110 IF K=0 THEN GOTO 320
120 IF K>N THEN PRINT:PRINT "k > n - functii nedefinite":GOTO 290
130 REM ---- k! --------------------------------------------------
140 LET R=K
150 GOSUB 380
160 LET A1=F
170 REM ---- (n-k)! ---------------------------------------------
180 LET R=N-K
190 GOSUB 380
200 LET A3=F
210 LET A=A2/A3
220 LET C=A2/(A1*A3)
230 PRINT: PRINT "A ("; N; ","; K; ") ="; A
240 PRINT: PRINT "C ("; N; ","; K; ") ="; C
250 LET P=A2
260 PRINT : PRINT "P ("; N; ") ="; P
270 GOTO 290
```

```
280 PRINT : PRINT "Valoare imposibila pentru  n (sau k) !"
290 PRINT : PRINT "Dati alte valori  (1=DA, 0=NU) ?": INPUT N
300 IF N=1 THEN CLS : GOTO 10
310 STOP
320 A=1
330 C=1
340 GOTO 230
349 REM
350 REM ----------------------------------------------------------
360 REM - R!  ----------------------------------------------------
370 REM ----------------------------------------------------------
380 LET F=1
390 FOR L=1 TO R
400   LET F=F*L
410 NEXT L
420 RETURN
```

```
1    REM  Programul 8b : COMBI2
2    REM  ******************************************************
10   REM Programul calculeaza C(n,k) pentru n,k dati
11   REM  ******************************************************
20   INPUT "n=";N
30   INPUT "k=";K
40   IF N<0 OR K<0 THEN GOTO 300
50   IF N<> INT(N) THEN GOTO 300
60   IF K<> INT(K) THEN GOTO 300
70   IF N=0 OR K>N THEN GOTO 400
80   LET  C=1
90   FOR J=1 TO K
100       LET C=C*(N-J+1)/J
110  NEXT J
120  PRINT "C("; N; ","; K; ")="; C
130  STOP
299  REM
300  PRINT "Valori incorecte pentru n sau k"
310  PRINT "n="; N
```

```
320  PRINT "k="; K
330  STOP
399  REM
400  PRINT "Functie nedefinita"
410  GOTO 310
```

```
1    REM Programul 8c : COMBI3
2    REM ****************************************************
10   REM         Triunghiul lui Pascal
11   REM Se tipareste numai  C(n,k) pentru n,k date
12   REM ****************************************************
30   INPUT "n="; N
40   INPUT "k="; K
44   IF N<2 THEN GOTO 200
45   IF K<0 OR K>N THEN GOTO 200
50   DIM V(N+1):   DIM N(N+1)
55   CLS
60   FOR J=2 TO N+1
62       LET V(J)=0
64   NEXT J
66   LET V(1)=1:  LET N(1)=1
80   FOR I = 2 TO N
90       FOR J=2 TO I
95           LET N(J)=V(J)+V(J-1)
100      NEXT J
110      FOR J=1 TO I
115          LET V(J)=N(J)
120      NEXT J
130  NEXT I
140  FOR J=2 TO K+1
145      LET N(J)=V(J)+V(J-1)
150  NEXT J
159  PRINT
160  PRINT "C("; N; ","; K; ")="; N(K+1) : PRINT
170  STOP
200  PRINT "Date gresite"
210  PRINT "n=";N; "  k=";K
220  GOTO 30
```

```
1    REM  Programul 9:    MATRICE
2    REM  ***********************************************************
10   REM  Se dau matricele A si B
11   REM  Pt. Kod=1 se calculeaza  C = A + B
12   REM  Pt. Kod=2 se calculeaza  C = A * B
13   REM  ***********************************************************
20   INPUT "Kod="; KOD
25   INPUT "Nr.linii A=";M1
30   INPUT "Nr.coloane A=";N1
35   INPUT "Nr.linii B=";M2
40   INPUT "Nr.coloane B=";N2
42   IF KOD=1 AND (M1<>M2 OR N1<>N2) THEN GOTO 300
43   IF KOD=2 AND (N1<>M2) THEN GOTO 300
45   DIM A(M1,N1), B(M2,N2), C(M1,N2)
50   FOR I=1 TO M1
55      FOR J=1 TO N1
60           PRINT "A(";I;",";J;")=";: INPUT A(I,J)
65      NEXT J
70   NEXT I
75   PRINT
80   FOR I=1 TO M2
85      FOR J=1 TO N2
90           PRINT "B(";I;",";J;")=";: INPUT B(I,J)
95      NEXT J
100  NEXT I
110  IF KOD=2 THEN GOTO 200
115  IF KOD<>1 THEN GOTO 500
120  REM Adunare
130  FOR I = 1 TO M1
140    FOR J = 1 TO N1
150       LET C(I,J) = A(I,J) + B(I,J)
160    NEXT J
170  NEXT I
180  GOTO 400
200  REM Produs
210  FOR I= 1 TO M1
220     FOR J=1 TO N2
230       LET C(I,J)=0
240       FOR K=1 TO N1
250           LET C(I,J)=C(I,J)+A(I,K)*B(K,J)
260       NEXT K
270     NEXT J
280  NEXT I
```

```
290  GOTO 400
300  PRINT: PRINT "Date gresite"
310  PRINT "Matricea A are"
312  PRINT M1;" linii"
314  PRINT N1;" coloane"
320  PRINT:PRINT "Matricea B are"
322  PRINT M2;" linii"
324  PRINT N2;"coloane"
330  STOP
400  REM  Tiparirea rezultatelor
405  PRINT:PRINT "Matricea C este:"
410  FOR I=1 TO M1
420    FOR J=1 TO N2
430        PRINT "C(";I; ","; J; ")="; C(I,J)
440    NEXT J
450  NEXT I
460  STOP
500  PRINT "Valoarea lui Kod e gresita"
510  PRINT "  Kod = "; KOD
520  STOP
```

```
1    REM Programul 10 :  NRE
2    REM ***********************************************************
10   REM Se tiparesc primii n termeni ai unui sir convergent la
13   REM                    numarul   e
14   REM ***********************************************************
20   INPUT "n="; N
30   LET A=1
40   LET T=1
50   FOR I=1 TO N
60       LET T=T/I
70       LET A=A+T
80       PRINT "a("; I; ")="; A
90   NEXT I
100  STOP
```

```
1   REM Programul 11a : FIBO1
2   REM *************************************************************
10  REM Acest program calculeaza numerele lui Fibonacci
14  REM          cu indicele intre I1 si I2
20  REM *************************************************************
60  INPUT "I1=";I1
62  INPUT "I2=";I2
66  IF I1<3 THEN GOTO 200
70  IF I1>I2 OR I1<>INT (I1) OR I2<>INT (I2) THEN GOTO 200
75  CLS
90  DIM F(I2+1)
100 LET F(1)=0
110 LET F(2)=1
120 FOR I=3 TO I1
130 LET F(I)=F(I-1)+F(I-2)
140 NEXT I
150 FOR I=I1+1 TO I2+1
160 LET F(I)=F(I-1)+F(I-2)
170 PRINT "F(";I-1;")=";F(I)
180 NEXT I: PRINT
185 IF I2>35 THEN  PRINT "Valori exacte sint numai pina la
                   termenul de indice 35"
190 STOP
200 PRINT "Valori gresite"
210 PRINT "I1=";I1;"   I2=";I2
220 PRINT "Trebuie ca I1>=3 si I1<=I2
230 GOTO 60
```

```
1   REM Programul 11b : FIBO1
2   REM *************************************************************
10  REM Acest program calculeaza numerele lui Fibonacci
14  REM          cu indicele intre I1 si I2
20  REM *************************************************************
60  INPUT "I1=";I1
62  INPUT "I2=";I2
66  IF I1<3 THEN GOTO 200
70  IF I1>I2 OR I1<>INT (I1) OR I2<>INT (I2) THEN GOTO 200
75  CLS
90  DIM F(I2+1)
```

```
100 LET F(1)=0
110 LET F(2)=1
120 FOR I=3 TO I1
130 LET F(I)=F(I-1)+F(I-2)
140 NEXT I  150 FOR I=I1+1 TO I2+1
160 LET F(I)=F(I-1)+F(I-2)
170 PRINT "F(";I-1;")=";F(I)
180 NEXT I: PRINT
185 IF I2>35 THEN PRINT "Valori exacte sint numai pina la
                          termenul de indice 35"
190 STOP
200 PRINT "Valori gresite"
210 PRINT "I1=";I1;"  I2=";I2
220 PRINT "Trebuie ca I1>=3 si I1<=I2
230 GOTO 60
```

```
1  REM Programul 12a :  RADICAL1
10 REM ***********************************************************
12 REM Acest  program  calculeaza  o valoare aproximativa
16 REM pentru radicalul de ordin 2 dintr-o valoare data A
22 REM ***********************************************************
70 DIM X(100)
80 INPUT "A="; A
90 IF A<0 THEN CLS:PRINT"Valoare negativa, dati alta ":GOTO 80
100 INPUT "EPS="; EP
110 LET X(1)=A
120 LET I=2
130 LET X(I)=(X(I-1)+A/X(I-1))/2
140 IF ABS (X(I)-X(I-1)) < EP THEN GOTO 170
150 LET I=I+1
160 GOTO 130
170 PRINT : PRINT "Radical de ordin 2 din ";A;" este";
175 PRINT "    r = ";X(I)
180 PRINT "S-au efectuat ";I-1;" iteratii"
190 PRINT: STOP
```

```
1  REM Programul 12b :  RADICAL2
10 REM **************************************************************
12 REM Acest program calculeaza o valoare aproximativa pentru
14 REM      radicalul de ordin 2 dintr-o valoare data A
22 REM **************************************************************
70 LET I=1
80 INPUT   "A="; A
90 IF A<0 THEN CLS :PRINT "Valoare negativa, dati alta":GOTO 80
100 INPUT "EPS=";EP
110 LET XV=A
120 LET XN= (XV+A/XV)/2  130 IF ABS (XN-XV) < EP THEN GOTO 170
140 LET XV=XN
150 LET I=I+1
160 GOTO 120
170 PRINT:PRINT "Radical de ordin 2 din ";A;" este";
175 PRINT "    r="; XN
180 PRINT "S-au efectuat ";I" iteratii": PRINT
190 STOP
```

```
1  REM Programul 12c :  RADICALN
10 REM **************************************************************
12 REM Acest program calculeaza o valoare aproximativa pentru
16 REM      radicalul de ordin n dintr-o valoare data A
22 REM **************************************************************
60 INPUT "n=";N
70 LET I=1
80 INPUT   "A="; A
90 IF A<0 THEN CLS : PRINT "Valoare negativa, dati alta":GOTO 80
100 INPUT "EPS=";EP
110 LET XV=A
120 LET XN= ( (N-1)*XV+A/XV^(N-1) )/N
130 IF ABS (XN-XV) < EP THEN GOTO 170
140 LET XV=XN
150 LET I=I+1
160 GOTO 120
170 CLS
172 PRINT : PRINT "Radical de ordin n din ";A;" este" ;
175 PRINT "    r="; XN
180 PRINT "S-au efectuat ";I" iteratii" : PRINT
190 STOP
```

```
1   REM Programul 13 : COARDATG
2   REM ********************************************************
10  REM Cu acest program se afla radacina ecuatiei F(x)=0
11  REM          din intervalul [a,b]
12  REM F se def.in linia 30, derivatele I si II in 40 si  50.
13  REM ********************************************************
30  DEF FN F(X)=X^2-3
40  DEF FN D(X)=2*X
50  DEF FN S(X)=2
100 INPUT "a=";A
110 INPUT "b=";B
120 IF FNF(A)=0 THEN PRINT "Ecuatia are ca radacina pe",A:STOP
130 IF FNF(B)=0 THEN PRINT "Ecuatia are ca radacina pe",B:STOP
140 INPUT "Precizia EPS=";EP
150 IF FNF(A)*FNF(B)>0 THEN
      PRINT "Ec. nu are o singura radacina in [a,b]": STOP
155 LET R=(A+B)/2
160 IF FND(R)*FNS(R) > 0 THEN LET C=A: LET T=B: GOTO 180
170 LET C=B:  LET T=A
180 LET T = T - FNF(T)/FND(T)
190 LET C = C - (T-C)*FNF(C) / (FNF(T)-FNF(C))
200 PRINT "c="; C;  "    t="; T
210 IF ABS(C-T)>=EP THEN GOTO 180
220 LET R=(C+T)/2
230 PRINT "Radacina este:";R
```

```
1    REM  Programul 14 : TANGENTE
2    REM  ******************************************************
10   REM  Din punctul A(XA,YA) se duc tangentele AP si AQ la
11   REM  cercul de centru O si raza r.
12   REM  Se presupune ca cercul poate fi desenat pe ecran.
13   REM  ******************************************************
20    INPUT "XA=";XA
22    INPUT "YA=";YA
24    INPUT "XO=";XO
26    INPUT "YO=";YO
28    INPUT "R="; R
```

```
30  LET XM=(XO+XA)/2
32  LET YM=(YO+YA)/2
40  LET D1=XM-XO:  LET D2=YM-YO
44  LET R1=SQR( D1*D1+D2*D2 )
50  SCREEN 1,0 :CLS
60  CIRCLE (XO,199-YO),R
70  PSET (XO,199-YO): PSET (XA,199-YA)
85  LET MD=(XO-XA)*(XO-XA)+(YO-YA)*(YO-YA)
86  IF MD<=R*R THEN GOTO 500
90  IF YO=YM THEN GOTO 200
100 LET C1=(XO-XM)/(YM-YO)
110 LET C2=(R*R-R1*R1)/(YM-YO)
120 LET C3=C2-C1*(XO+XM)/(YM-YO)
122 LET C3=C3/2
130 LET MA=1+C1*C1
135 LET MB=XO-C1*C3
140 LET MC=C3*C3+XO*XO-R*R
145 LET MD=MB*MB-MA*MC
150 LET MR=SQR(MD)
160 LET X1=(MB+MR)/MA  165  LET X2=(MB-MR)/MA
170 LET Y1=YO+C1*X1+C3
175 LET Y2=YO+C1*X2+C3
180 GOTO 250
200 LET X1=XO*XO-XM*XM+R1*R1-R*R
205 LET X1/(XO-XM)/2
210 LET X2=X1
220 LET D=X1-XO: LET MD=R*R-D*D
225 LET MR=SQR(MD)
230 LET Y1=YO-MR
235 LET Y2=YO+MR
250 LET A$="P"
260 GOSUB 300
270 LET X1=X2:      LET Y1=Y2
280 LET A$="Q"
285 GOSUB 300
290 STOP
300 REM Se traseaza o tangenta
320 PSET (X1,199-Y1)
330 LINE (XO,199-YO)-(X1,199-Y1)
340 RETURN
500 PRINT "A este interior cercului"
```

```
1   REM Programul 15 : GRAFIC
2   REM ************************************************************
10  REM Deseneaza graficul unei functii
11  REM ************************************************************
15  DEF FN F(X)=SIN(X*X)
20  DEF FN G(X)=INT((X-A)*319/(B-A)+.5)
30  DEF FN H(Y)=INT((Y-C)*190/(D-C)+.5)
40  CLS:PRINT "Programul deseneaza graficul functiei definite
        in linia 15"
50  PRINT "Urmeaza sa se precizeze datele initiale: a,b,c,d"
60  INPUT "a=";A:INPUT "b=";B
70  IF A<B THEN GOTO 90
80  PRINT "Eroare: trebuie ca a<b" : GOTO 60
90  INPUT "c=";C: INPUT "d=";D
100  IF C<D THEN GOTO 120
110  PRINT "Eroare: trebuie ca c<d" : GOTO 90
120  INPUT "Pasul pentru desenare, in nr.puncte ecran:";H
130  IF H>0 AND H<319 AND H=INT(H) THEN GOTO 150
140  PRINT "Eroare: Pasul trebuie sa fie intreg si cuprins intre
        1 si 319." : GOTO 60
150  LET H1=H*(B-A)/319
160  LET X=A
170  LET Y=FN F(X)
180  IF Y<C OR Y>D THEN  GOTO 380  190  LET X1=0
200  LET Y1=FN H(Y)
210  SCREEN 1,0 : CLS
220  LET X2=X1: LET Y2=Y1
230  LET X=X+H1
240  IF X>B THEN GOTO 310
250  LET Y=FN F(X)
260  IF Y<C OR Y>D THEN GOTO 380
270  LET X1=FN G(X)
280  LET Y1= FN H(Y)
290  LINE(X1,190-Y1)-(X2,190-Y2)
300  GOTO 220
310  IF A*B>0 THEN GOTO 340
320  LET X=0: LET X1=FN G(X)
330  LINE (X1,0)-(X1,190)
340  IF C*D>0 THEN GOTO 370
350  LET Y=0:   LET Y1=FN H(Y)
```

```
360  LINE(0,190-Y1)-(319,190-Y1)
370  STOP
380  PRINT "Eroare: Valorile functiei depasesc intervalul dat"
390  END
```

```
1  REM  Programul 16 : CONICE
2  REM  ***********************************************************
10 LET A1=-159:  LET A2=159: LET B1=-99: LET B2=99: LET J=1
11 LET H=1
20 DEF FN G(X)=INT((X-A1)*319/(A2-A1)+.5)
30 DEF FN H(Y)=INT((Y-B1)*199/(B2-B1)+.5)
40 DEF FN U(X)=B*SQR(A*A-X*X)/A
50 DEF FN V(X)=B*SQR(X*X-A*A)/A
60 DEF FN W(X)=SQR(2*P*X)
70 CLS: PRINT
80 PRINT "Programul ofera functiile:": PRINT
90 PRINT "  1. Afisarea sau modificarea conditiilor de lucru."
95 PRINT
100 PRINT "  2. Desenarea unei elipse.": PRINT
110 PRINT "  3. Desenarea unei hiperbole.": PRINT
120 PRINT "  4. Desenarea unei parabole.": PRINT
130 PRINT "  5. Oprirea programului.": PRINT
140 PRINT "Ce varianta alegeti(1-5)?"
150 LET R$=INKEY$
160 IF R$="1" THEN GOTO 220
170 IF R$="2" THEN GOTO 400
180 IF R$="3" THEN GOTO 500
190 IF R$="4" THEN GOTO 530
200 IF R$="5" THEN STOP
210 GOTO 150
220 CLS:
 PRINT"Conditiile de lucru stabilite pina in acest moment sint:"
230 PRINT"-pe ecranul calculatorului se reprezinta";"
  " dreptunghiul din plan pentru care:"
240 PRINT:PRINT "        ";A1;"<=x<=";A2
250 PRINT:PRINT "        ";B1;"<=y<=";B2
260 PRINT:PRINT"-pasul pentru desenarea conicelor, in numar";
  " puncte ecran, este egal cu:";j
270 INPUT "Doriti modificarea acestor valori(D,N)?";R$
```

```
280 IF R$="D" OR R$="d" THEN GOTO 310
290 IF R$="N" OR R$="n" THEN GOTO 70
300 GOTO 270
310 CLS
315 PRINT "Noile limite pentru x: a1 si a2:"
320 INPUT "a1=";A1:INPUT"a2=";A2
330 IF A1>=A2 THEN PRINT "Valori eronate!": GOTO 315
340 PRINT "Noile limite pentru y: b1 si b2:"
350 INPUT "b1=";B1:INPUT "b2=";B2
360 IF B1>=B2 THEN PRINT "Valori eronate!": GOTO 350
370 INPUT "Pasul in nr.puncte ecran ";J
380 IF J<=0 THEN PRINT "Valoare eronata!": GOTO 370
390 LET H=(A2-A1)*J/320: GOTO 220
400 CLS:LET T=1: LET S$="+"
410 PRINT:PRINT "Ecuatia unei elipse este de forma:":PRINT: PRINT
420 PRINT "       2       2 "
430 PRINT "      x       y"
440 PRINT "     ---- ";S$;" ---- = 1"
450 PRINT "       2       2 "
460 PRINT "      a       b "
470 PRINT: PRINT:PRINT "Care sint valorile pentru a si b?"
480 INPUT "a=";A:INPUT "b=";B: GOTO 600
500 CLS: LET T=2: LET S$="-"
510 PRINT: PRINT "Ecuatia unei hiperbole este de forma:"
511 PRINT: PRINT
520 GOTO 420
530 CLS: LET T=3
540 PRINT: PRINT "Ecuatia unei parabole este de forma:"
541 PRINT: PRINT
550 PRINT "      2": PRINT "     y = 2*p*x"
560 PRINT: PRINT: PRINT "Care este valoarea lui p?"
570 INPUT "p=";P
600 SCREEN 1,0:CLS: LET R=1
610 LET I=1
620 IF T<>1 THEN GOTO 670
630 LET C=A1: LET D=A2
640 IF C<-A THEN LET C=-A
650 IF D>A THEN LET D=A
660 GOTO 730
670 IF T<>2 THEN GOTO 710
680 LET C=A1: LET D=A2
```

```
690 IF D>-A THEN LET D=-A
700 GOTO 730
710 LET C=A1: LET D=A2 720 IF C>0 THEN LET C=0
730 LET X=C: LET K=0
740 IF X>D THEN GOTO 850
750 IF T=1 THEN LET Y=FN U(X): GOTO 780
760 IF T=2 THEN LET Y=FN V(X): GOTO 780
770 LET Y=FN W(X)
780 IF R=2 THEN LET Y=-Y
790 IF Y<B1 OR Y>B2 THEN LET K=0: GOTO 840
800 LET X1=FN G(X): LET Y1=FN H(Y)
810 IF K=0 THEN PSET(X1,200-Y1): LET K=1: GOTO 830
820 LINE (X1,200-Y1)-(X2,200-Y2)
830 LET X2=X1: LET Y2=Y1
840 LET X=X+H: GOTO 740
850 IF X<D+H THEN LET X=D: GOTO 740
855 IF T<>2 THEN GOTO 900
860 IF I=2 THEN GOTO 900
870 LET C=A1: LET D=A2: LET I=2
880 IF C<A THEN LET C=A
890 GOTO 730
900 LET R=R+1: IF R=2 THEN GOTO 610
910 IF B1*B2>0 THEN GOTO 940
920 LET Y=0: LET Y1=FN H(Y)
930 LINE (0,200-Y1)-(320,200-Y1)
940 IF A1*A2>0 THEN GOTO 1000
950 LET X=0: LET X1=FN G(X)
960 LINE (X1,0)-(X1,200)
1000 INPUT "Pt.continuare tastati ENTER ";I$: GOTO 70
```

```
1    REM Programul 17 :  MEDISP
2    REM ***************************************************
10   REM Calculul mediei si dispersiei
11   REM ***************************************************
15   REM ----------- citirea datelor
20   INPUT "n ="; N  : DIM  X(N)
```

```
30  FOR I = 1 TO N
40      PRINT "x("; I; ")=";: INPUT X(I)
50  NEXT I
55  REM ----------- calculul mediei
60  LET M = 0
70  FOR I = 1 TO N
80      LET M = M + X(I)
90  NEXT I
100 LET  M = M / N
105 REM ----------- calculul dispersiei
110 LET D = 0  120  FOR I = 1 TO N
130     LET D = D + ( X(I) - M ) * ( X(I) - M )
140 NEXT I
150 LET D = D / N
155 REM ---------- tiparirea datelor si a rezultatelor
160 CLS : FOR I = 1 TO N
170     PRINT "x("; I; ")="; X(I)
180 NEXT I  : PRINT
190 PRINT "media = ";  M
200 PRINT "dispersia = ";  D
210 PRINT : STOP
```

```
1   REM Programul 18 HISTOGRAMA
2   REM ******************************************************
10  REM Centralizare + histograma
11  REM ******************************************************
15  REM ---------- citirea datelor
16  CLS
20  INPUT "Numarul datelor:"; N
30  DIM X(N)
40  FOR I=1 TO N
50      PRINT "x("; I; ")=";:INPUT X(I)
60  NEXT I
65  REM ---------- calculul minimului si maximului
70  LET M = INT (1+1.442726 * LOG(N)) : DIM A(M+1), F(M)
80  LET M1 = X(1) : LET M2 = X(1)
90  IF N = 1 THEN GOTO 150
```

```
100  FOR I = 2 TO N
110       IF M1 > X(I) THEN LET M1 = X(I)
120       IF M2 < X(I) THEN LET M2 = X(I)
130  NEXT I
140  IF M1 = M2 THEN LET M = 1
150  LET A(1) = M1 : LET A(M+1) = M2
160  LET P = ( A(M+1) - A(1) ) / M
170  FOR I = 1 TO M
180       LET F(I) = 0
190       LET A(I) = M1 + (I-1) * P
200  NEXT I
210  IF P = 0 THEN LET  F(1) = N : GOTO  270
220  FOR I = 1 TO N
230       LET J = INT ( (X(I)-M1)/P) + 1
240       IF J > M THEN LET J = M
250       LET F(J) = F(J) + 1
260  NEXT I
265  REM ---------- tiparire intervale + frecvente
270  PRINT "Intervalul"; "
280  FOR I = 1 TO M-1
290       PRINT "["; A(I); ","; A(I+1); ")"; "           "; F(I)
300  NEXT I
310  PRINT "["; A(M); ","; A(M+1); "]"; "           "; F(M)
313  PRINT:PRINT:PRINT  "Apasati orice tasta pentru continuare !"
314  IF INKEY$="" THEN GOTO 314
315  REM ---------- trasare histograma
318  SCREEN 1,0
320  CLS:  LET D = F(1)
330  IF M = 1 THEN  GOTO 370
340  FOR I = 2 TO M
350       IF D < F(I) THEN LET D = F(I)
360  NEXT I
370  LET X2 = 4 : LET E = A(M+1)-A(1)
380  LINE (0,190)-(319,190)
385 LET K=0
390  FOR I = 1 TO M
395 LET K=K+1: IF K>3 THEN K=1
400    LET X1 = X2 : LET X2 = 320
410    IF E<>0 THEN  LET X2 = X1 + INT ((A(I+1)-A(I))*310/E+.5)
420    LET Y = 10 + INT (F(I)*160/D + .5)
430     LINE (X1,195)-(X1,200-Y)
```

```
440    LINE (X1,200-Y)-(X2,200-Y): LINE(X2,200-Y)-(X2,190)
450    IF F(I) = 0 THEN  GOTO 500
460    LET J = X1 + 1
470    IF J >= X2  THEN  GOTO 500
480    LINE (J,190)-(J,200-Y),K
490    LET J=J+1   : GOTO 470
500  NEXT I
510  STOP
```

```
1    REM Programul 19 : MEDII
2    REM ************************************************************
10   REM Se ordoneaza elevii unei clase in ordinea
11   REM    descrescatoare a mediilor generale
12   REM ************************************************************
18   REM       M = nr.materiilor
19   REM       NR = nr.elevilor
20   REM       E$(i) - numele elevului "i"
21   REM       N(i,j) - nota elevului i la disciplina j
22   REM     la disciplina j
23   REM ************************************************************
25   READ NR,M
30   DIM E$(NR), N(NR,M), V(NR)
35   CLS
40   PRINT "Numele si prenumele   Media" : PRINT
50   FOR I=1 TO NR
55       LET S=0
60       READ E$(I)
65       FOR J=1 TO M
70           READ N(I,J)
80           LET S=S+N(I,J)
90       NEXT J
100      LET V(I)=S/M
110      PRINT E$(I);
120      PRINT "        "; V(I)
130  NEXT I
190  LET INV=0
```

```
200  FOR I = 1 TO NR-1
210      IF V(I)>=V(I+1) THEN GOTO 250
220      LET T=V(I)
221      LET V(I)=V(I+1)
222      LET V(I+1)=T
230      LET T$=E$(I)
231      LET E$(I)=E$(I+1)
232      LET E$(I+1)=T$
240      LET INV=INV+1
250  NEXT I
260  IF INV<>0 THEN GOTO 190
270  PRINT: PRINT
300  FOR I=1 TO NR
310      PRINT E$(I);"          "; V(I)
320  NEXT I : PRINT
400  DATA 3,14
410  DATA "Abrudan Adina"
411  DATA 9,8,5,8,7,7,6,7,10,9
412  DATA 8,9,10,10
420  DATA "Bidian Sorin"
421  DATA 8,6,9,5,8,7,10,8,8,9
422  DATA 9,8,10,10
430  DATA "Frentiu Alin"
431  DATA 8,8,9,9,8,7,10,8,8,9
432  DATA 9,8,10,10
440  STOP
```

```
1    REM Programul 20 : RESTURI
2    REM ***********************************************************
10   REM Operatii in Zn
12   REM ***********************************************************
99   REM
100  CLS : PRINT "Dati un intreg n, n>1"
101  PRINT "sau 0 pentru STOP"
110  INPUT "n = ";N
120  IF N=0 THEN STOP
130  IF N<=1 OR N<>INT (N) THEN GOTO 100
```

```
299 REM
300 CLS: PRINT "Va oferim citeva posibilitati de operare in Z";N
301 PRINT
310 PRINT " 1. Se da i din Z";N;" si se obtine clasa lui i din Z"
      ;N
320 PRINT " 2. Se dau i si j din Z";N;" si se calculeaza i+j si
      i*j in Z";N
330 PRINT " 3. Se da i din Z";N;" si se obtine -i si i^(-1) in Z"
      ;N
340 PRINT " 4. Se tiparesc perechile de divizori ai lui 0 in Z";N
350 PRINT " 5. Se tipareste tabla adunarii in Z";N
360 PRINT " 6. Se tipareste tabla inmultirii in Z";N
370 PRINT " 0. Se cere o noua valoare pentru n"
380 INPUT "Ce doriti (1,2,3,4,5,6,0) ";V
382 PRINT
599 REM
600 IF V=0 THEN GOTO 100
610 IF V=1 THEN GOTO 1000
620 IF V=2 THEN GOTO 1300
630 IF V=3 THEN GOTO 1600
640 IF V=4 THEN GOTO 1900
650 IF V=5 THEN GOTO 2200
660 IF V=6 THEN GOTO 2500
670 GOTO 300
1000 INPUT "i = ";I
1010 IF I<>INT (I) THEN GOTO 1000
1020 LET C=ABS (I)-INT (ABS (I)/N)*N: IF I<0 THEN LET C=N-C
1030 PRINT I;" (mod ";N;") = ";C
1040 GOTO 4000
1299 REM
1300 INPUT "i = ";I: INPUT "j = ";J
1310 IF I<0 OR I>=N OR I<>INT (I) OR J<0 OR J>=N OR J<>INT (J)
     THEN GOTO 1300
1320 PRINT I;" + ";J;" (mod ";N;") = ";I+J-INT ((I+J)/N)*N
1330 PRINT I;" * ";J;" (mod ";N;") = ";I*J-INT ((I*J)/N)*N
1340 GOTO 4000
1599 REM
1600 INPUT "i = ";I
1610 IF I<0 OR I>=N OR I<>INT (I) THEN GOTO 1600
1620 PRINT "Elementul opus lui ";I;" in Z";N; "este "; N-I
1630 PRINT "Elementul invers lui ";I;" in Z";N;" ";
```

```
1640 FOR J=1 TO N-1
1650   LET C=I*J-INT ((I*J)/N)*N
1660   IF C<>1 THEN GOTO 1690
1670   PRINT "este ";J
1680   GOTO 4000
1690 NEXT J  1700 PRINT "nu exista"
1710 GOTO 4000
1899 REM
1900 PRINT "Divizori ai lui 0 in Z";N
1910 LET K=0
1920 FOR I=1 TO N-1
1930   LET C=I*I-INT ((I*I)/N)*N
1940   IF C<>0 THEN GOTO 1960
1950   LET K=K+1: PRINT I;" * ";I;" = 0"
1960   FOR J=I+1 TO N-1
1970     LET C=I*J-INT ((I*J)/N)*N
1980     IF C<>0 THEN GOTO 2000
1990     LET K=K+1: PRINT I;" * ";J;" = 0 si ";J;" * ";I;" = 0"
2000   NEXT J
2010 NEXT I
2020 IF K=0 THEN PRINT "  nu exista"
2030 GOTO 4000
2199 REM
2200 PRINT "Tabla adunarii in Z";N: PRINT " +  ";
2210 FOR J=0 TO N-1: PRINT J;" ";: NEXT J: PRINT
2220 FOR I=0 TO N-1
2230   PRINT: PRINT I;"  ";
2240   FOR J=0 TO N-1
2250     PRINT I+J-INT ((I+J)/N)*N;" ";
2260   NEXT J
2270 NEXT I
2280 GOTO 4000
2499 REM
2500 PRINT "Tabla inmultirii in Z";N: PRINT "   ";
2510 FOR J=0 TO N-1: PRINT J;" ";: NEXT J: PRINT
2520 FOR I=0 TO N-1
2530 PRINT: PRINT I;"  ";
2540 FOR J=0 TO N-1
2550 PRINT I*J-INT ((I*J)/N)*N;" ";
2560 NEXT J
2570 NEXT I
```

```
2580 GOTO 4000
3999 REM
4000 PRINT: PRINT :PRINT: PRINT "Apasati orice tasta"
4010 IF INKEY$="" THEN GOTO 4010
4020 GOTO 300
```

```
1    REM Programul 21 : RIEMANN
2    REM ****************************************************
3    REM Pentru [a,b] dat si functia f data se calculeaza suma
4    REM    Riemann
5    REM
5    REM
6    DEF FN F(X)=4/(X*X+1)
7    DEF FN S(I)=( X(I)+X(I+1) ) / 2
8    PRINT "Definiti functia f(x) in linia 6. "
9    PRINT "Daca ati facut-o tastati 0, altfel orice"
10   INPUT I$ :IF I$="0" THEN GOTO 15 ELSE STOP
15   PRINT "Daca pentru suma Riemann luati ca puncte";
          "(x(i)+x(i+1))/2 tastati 0"
16   INPUT J$ :IF J$="0" THEN GOTO 20 ELSE STOP
20   LET MAXN=500 : DIM X(MAXN), P(MAXN)
30   CLS: PRINT "Se da un interval de numere reale [a,b]"
35   PRINT "(Daca a=0 si b=0 atunci STOP)"
40   INPUT "a = ";A
50   INPUT "b = ";B
60   IF A=0 AND B=0 THEN STOP
70   IF A>=B THEN GOTO 30
80   LET X(1)=A
99   REM
100  CLS: PRINT
     "Va oferim doua moduri de definire a unei diviziuni peste";
105  PRINT  "[ ";A;" , ";B;" ]"
110  PRINT " 1. Alegerea de catre Dv. a nodurilor intermediare"
120  PRINT " 2. Definirea unei diviziuni echidistante"
130  INPUT "Care varianta o alegeti (1,2) ";V
140  IF V<>1 AND V<>2 THEN GOTO 100
200  IF V=2 THEN GOTO 300
210  LET N=1: LET XS=A
```

```
220  PRINT "x(";N+1;") din (";XS;" , ";B;"] = ?";: INPUT X(N+1)
230  IF X(N+1)=B THEN GOTO 1000
240  IF X(N+1)<=XS OR X(N+1)>B THEN GOTO 220
250  LET N=N+1: LET XS=X(N): IF N>MAXN-1 THEN GOTO 1000
260  GOTO 220
299  REM
300  PRINT " 3. Prin numarul de subintervale"
310  PRINT " 4. Prin lungimea unui subinterval"
320  INPUT "Care varianta o alegeti (3,4) ";V
330  IF V<>3 AND V<>4 THEN GOTO 300
399  REM
400  IF V=4 THEN GOTO 500
410  INPUT "n = ";N
420  IF N<1 OR N>=MAXN OR N<>INT (N) THEN GOTO 410
430  LET H=(B-A)/N: GOTO 600
499  REM
500  INPUT "h = ";H
510  IF H<=0 OR H>=B-A THEN GOTO 500
520  LET N=INT ((B-A)/H)+1: IF N>=MAXN THEN GOTO 500
599  REM
600  FOR I=2 TO N: LET X(I)=A+(I-1)*H: NEXT I
610  LET X(N+1)=B
999  PRINT
1000 CLS: PRINT
 "In diviziune se poate alege un sistem de puncte intermediare:"
1010 PRINT " 5. Alegind Dv. fiecare punct"
1020 PRINT " 6. Considerind expresia P(i) de calcul al punctului
     din al i-lea subinterval"
1025 PRINT "     [x(i),x(i+1)], definita deja in linia 7"
1030 INPUT "Care varianta o alegeti (5,6) ";V
1040 IF V<>5 AND V<>6 THEN GOTO 1000
1199 REM
1200 IF V=6 THEN GOTO 1400
1210 FOR I=1 TO N
1220     PRINT "p(";I;") din [";X(I);",";X(I+1);"]=";:INPUT P(I)
1230     IF X(I)>P(I) OR P(I)>X(I+1) THEN GOTO 1220
1240 NEXT I
1250 GOTO 2000
1400 REM
```

```
1420 FOR I=1 TO N
1430    LET P(I)=FN S(I)
1440    IF P(I)<X(I) THEN LET P(I)=X(I)
1450    IF P(I)>X(I+1) THEN LET P(I)=X(I+1)
1460 NEXT I
1999 REM
2000 REM
2020 LET NORMA=0: LET SUMA=0
2030 CLS: PRINT "Pentru intervalul [";A;",";B;"] si functia data";
2031 PRINT " avem diviziunea si punctele:"
2040 FOR I=1 TO N
2050    PRINT " [";X(I);" , ";X(I+1);"] contine ";P(I)
2060    LET HI=X(I+1)-X(I): IF HI>NORMA THEN LET NORMA=HI
2070    LET SUMA=SUMA+FN F(P(I))*HI
2075    IF INT(I/21)*21<>I THEN GOTO 2080
2076    PRINT "                                          Apasati
    orice tasta!"
2077    IF INKEY$="" THEN GOTO 2077
2080 NEXT I
2499 REM
2500 PRINT : PRINT "Norma diviziunii este ";NORMA
2510 PRINT "Valoarea sumei Riemann este ";
2511 PRINT "  Suma ="; SUMA
2515 PRINT "                                          Apasati
    orice tasta!"
2520 IF INKEY$="" THEN GOTO 2520
2530 GOTO 30
```

```
1   REM Programul 22 : APROXINT
2   REM ************************************************************
3   REM Calculul aproximativ al integralelor prin metoda
5   REM               dreptunghiurilor
6   REM
10  DEF FN F(X) = X*X
100 CLS:PRINT
    "Calcul aprox. al integralelor; metoda dreptunghiurilor"
110 PRINT: PRINT
120 PRINT "Definiti functia F(x) in linia 10"
```

```
123 PRINT "Daca ati facut-o tastati 0, altfel orice "
125 INPUT I$: IF I$="0" THEN GOTO 130 ELSE STOP
130 INPUT "a =";A
140 INPUT "b =";B
150 INPUT "epsilon =";EPS
159 REM
160 LET H=B-A
170 LET SD=FN F(B)
180 LET DV=H*SD
190 LET N=1
199 REM
200 LET NOD=A+H/2
209 REM
210 FOR K=1 TO N
220    LET SD=SD+FN F(NOD)
230    LET NOD=NOD+H
240 NEXT K
249 REM
250 LET H=H/2
260 LET DN=H*SD
270 LET N=2*N
279 REM
280 PRINT N;" dreptunghiuri ==> "
290 PRINT "          ";DN
299 REM
300 IF ABS (DN-DV)<EPS THEN GOTO 330
309 REM
310 LET DV=DN
320 GOTO 200
329 REM
330 PRINT "Integrala de la ";A
340 PRINT "          la ";B
350 PRINT "din functia data"
360 PRINT "cu precizia       ";EPS
370 PRINT "este              ";DN
380 PRINT
390 PRINT "S-au folosit ";N
400 PRINT "dreptunghiuri"
410 STOP
600 CLS:PRINT "Calcul aprox. al integralelor; metoda trapezelor"
610 DEF FN F(X) = X*X
```

```
620  PRINT "Definiti functia F(x) in linia 610"
625  PRINT "Daca ati facut-o tastati 0"
627  INPUT I$: IF I$="0" THEN GOTO 630 ELSE STOP
630  INPUT "a =";A
640  INPUT "b =";B
650  INPUT "epsilon =";EPS
659  REM
660  LET H=B-A
670  LET FAB=(FN F(A)+FN F(B))/2: LET ST=0
680  LET TV=H*FAB
690  LET N=1
699  REM
700  LET NOD=A+H/2
709  REM
710  FOR K=1 TO N
720     LET ST=ST+FN F(NOD)
730     LET NOD=NOD+H
740  NEXT K
749  REM
750  LET H=H/2
760  LET TN=H*ST
770  LET N=2*N
779  REM
780  PRINT N;" trapeze ==> "
790  PRINT "              ";TN
799  REM
800  IF ABS (TN-TV)<EPS THEN GOTO 830
809  REM
810  LET TV=TN
820  GOTO 700
830  PRINT "Integrala de la ";A
840  PRINT "             la ";B
850  PRINT "din functia data"
860  PRINT "cu precizia       ";EPS
870  PRINT "este              ";TN
880  PRINT
890  PRINT "S-au folosit ";N
900  PRINT "trapeze"
910  STOP
```

```
1    REM  Programul 23 : APLICINT
2    REM ****************************************************
10   DEF FN F(X)= X
20   DEF FN P(X)= 1
90   LET PI=4*ATN(1)
100  CLS: PRINT "Pentru o functie continua:"
110  PRINT "        f:[a,b]--->R": PRINT
120  PRINT "si pentru domeniul delimitat de:"
130  PRINT "  x=a, x=b, y=0, y=f(x)": PRINT
140  PRINT "va putem ajuta sa calculati:"
142  PRINT "arie, volum, lungime grafic, arie laterala,";
          " centru de greutate"
150  PRINT:PRINT
152  PRINT "Definiti functia F(x) in linia 10, iar F'(x) ";
          "( notat cu P(x) ) in li
154  PRINT "Daca ati facut-o tastati 0, altfel orice"
156  INPUT  I$: IF I$ ="0" THEN GOTO 158 ELSE STOP
158  CLS
160  PRINT "1. Aria domeniului"
170  PRINT "2. Volumul corpului de rotatie (in jurul lui y=0)"
180  PRINT "3. Lungimea graficului y=f(x) (daca f' continua)"
190  PRINT "4. Aria suprafetei laterale de rotatie (in jurul ";
          "lui y=0, daca f' contiunua"
200  PRINT "5. Coordonatele Xg si Yg ale centrului de greutate";
          "(daca f' > 0)"
210  PRINT
220  INPUT "Ce doriti? (1,2,3,4,5 sau 0 pentru STOP)";V
230  IF V=0 THEN STOP
499  REM
500  CLS
510  INPUT "a =";A
520  INPUT "b =";B
999  REM
1000 IF V<>1 THEN GOTO 1100
1020 GOSUB 4000
1030 GOSUB 6000
1040 PRINT "Aria domeniului =": PRINT "  ";TN
1050 GOTO 7000
1099 REM
1100 IF V<>2 THEN GOTO 1200
1120 GOSUB 4000
```

```
1130 GOSUB 6000
1140 PRINT "Volumul corpului =": PRINT " ";PI*TN
1150 GOTO 7000
1199 REM
1200 IF V<>3 THEN GOTO 1300
1220 GOSUB 4000
1230 GOSUB 6000
1240 PRINT "Lungimea graficului =": PRINT " ";TN
1250 GOTO 7000
1299 REM
1300 IF V<>4 THEN GOTO 1400
1330 GOSUB 4000
1340 GOSUB 6000
1350 PRINT "Aria laterala de rotatie =": PRINT " ";2*PI*TN
1360 GOTO 7000
1399 REM  1400 IF V<>5 THEN GOTO 158
1420 LET V=1: GOSUB 4000
1430 IF TN<.0000001 THEN GOSUB 6000: PRINT "f este nepozitiva":
   GOTO 7000
1440 LET ARIA=TN: LET V=2: GOSUB 4000
1450 LET YG=TN/ARIA/2: LET V=5
1460 GOSUB 4000
1470 GOSUB 6000
1480 PRINT " Xg = ";TN/ARIA: PRINT " Yg = ";YG
1490 GOTO 7000
1999 REM
2020 RETURN
2999 REM
3020 RETURN
3999 REM
4000 LET H=B-A
4010 LET X=A: GOSUB 5000: LET FAB=F: LET X=B: GOSUB 5000:
   LET FAB=(FAB+F)/2
4020 LET ST=0: LET TV=H*FAB: LET N=1
4030 LET NOD=A+H/2
4040 FOR K=1 TO N
4050    LET X=NOD: GOSUB 5000: LET ST=ST+F
4060    LET NOD=NOD+H
4070 NEXT K
4080 LET H=H/2
4090 LET TN=H*(FAB+ST)
```

```
4100 LET N=2*N
4110 LET TV=ABS (TN-TV): IF TV<.01 THEN RETURN
4120 PRINT AT 0,0;TV: LET TV=TN
4130 GOTO 4030
4999 REM
5000 REM Alegerea functiei de sub integrala
5010 IF V=1 THEN LET F=FN F(X): RETURN
5020 IF V=2 THEN LET F=FN F(X)*FN F(X): RETURN
5030 IF V=3 THEN LET F=SQR (1+FN P(X)*FN P(X)): RETURN
5040 IF V=4 THEN LET F=FN F(X)*SQR (1+FN P(X)*FN P(X)): RETURN
5050 IF V=5 THEN LET F=X*FN F(X): RETURN
5999 REM
6000 CLS
6010 PRINT "Pentru:": PRINT
6020 PRINT " a = ";A
6030 PRINT " b =";B
6070 PRINT: PRINT "====>": PRINT
6080 RETURN
6999 REM
7000 PRINT:PRINT:PRINT:PRINT:
     PRINT "Apasati orice tasta pentru continuare!"
7005 IF INKEY$="" THEN GOTO 7005
7010 GOTO 158
```

B I B L I O G R A F I E

1. Coman, Gh., Frenţiu M., Introducere în informatică, Editura Dacia, Cluj-Napoca, 1982.

2. Dodescu, Gh., Ionescu, D., Nisipeanu, L., Pilat, F., Limbajul Basic şi aplicaţii, Ed.didactică şi pedagogică, Bucureşti 1978.

3. Dumitraşcu, H., Să învăţăm Basic,Editura Albatros, Bucureşti, 1987.

4. Frenţiu, M., Kasa, Z., Tarţia, C., Ţîmbulea, L., Utilizarea calculatorului personal PRAE-M, Universitatea Cluj-Napoca, 1986.

5. Kasa, Z., Ismerkedes az informatikaval, Editura Dacia, Cluj-Napoca 1983.

7. * * * Biblioteca MATH I, ITCI Filiala Cluj-Napoca, 1987.

8. * * * Buletinul roman de Informare Tehnică - BIT - nr.5/1986.

9. * * * Manuale şcolare.

CUPRINS

microinformatica

Vreţi să ştiţi cum să utilizaţi calculatorul PC-XT şi AT şi să-i folosiţi la maxim calităţile ţinînd seama de limitele lui?

Curiozitatea dumneavoastră a fost anticipată de MicroInformatica, firma ce deja a cucerit atenţia a zeci de mii de utilizatori, prin titlurile de carte alese strict după necesităţi, prin frecvenţa mare de apariţie a cărţilor şi memento-urilor, prin preţurile deosebit de avantajoase, precum şi prin promptitudinea cu care intraţi în posesia lor.

În **seria manuale**, MicroInformatica vă propune lista manualelor disponibile în prezent (aprilie 1992):

- **Ce este şi ce vă oferă calculatorul IBM-PC**
  O descriere a calculatoarelor de tip IBM PC AT şi XT cu principalele lor domenii de utilizare precum şi cele mai utilizate produse program.

- **Sistemul de operare DOS - Comenzi**
  Descrierea comenzilor MS-DOS versiunile 3.30, 4.0 şi 5.0.

- **Sistemul de operare DOS - Funcţii sistem**
  Manualul oferă o descriere a funcţiilor disponibile prin întreruperea INT 21H, aşa numitele funcţii sistem.

- **Sistemul de operare DOS - Ghidul programatorului**
  Structura sistemului de operare, gestiunea tastaturii, a ecranului, a discului flexibil şi a discului Winchester, funcţiile BIOS.

- **FoxBase+ - Comenzi şi funcţii**
  Cartea prezintă într-un mod sistematic, în ordine alfabetică, toate comenzile şi funcţiile versiunii 2.10.

- **Reţele locale**
  O prezentare în acelaşi timp generală şi detaliată a unui subiect modern, încă insuficient cunoscut la noi: tipuri de reţele, organizarea lor, modul de lucru, posibilităţi oferite.

- **Ghid de utilizare Turbo C**
  Sînt descrise, pe bază de exemple clare, instrucţiunile limbajului, facilităţile oferite de mediul integrat, posibilităţile de interfaţare cu alte limbaje, anumite noţiuni de programare avansată.

- **Ghid de utilizare dBase III Plus**
  O trecere în revistă a setului complet de instrucţiuni şi funcţii pe bază de exemple şi prezentarea unei aplicaţii interesante.

- **Ghid de utilizare dBase IV**
  Particularităţile specifice ale produsului dBase IV, modurile de lucru cu interfeţele neprocedurale. Prezentarea SQL, QBE şi generatorului de aplicaţie recomandă cartea tuturor utilizatorilor de SGBD-uri.

- **Ghid de utilizare TURBO PASCAL 6.0**
  Prezentare completă a noului mediu de programare Turbo Pascal 6.0 al firmei Borland International cu noile sale facilităţi.

Seria Ghid de iniţiere se adresează cititorilor care au cunoştinţe minimale despre utilizarea unui calculator de tip IBM-PC. Lucrările sînt concepute sub forma unor "lecţii şedinţe" şi presupun că cititorul are acces la calculator concomitent cu parcurgerea conţinutului lor.

- **Ghid de iniţiere WORDPERFECT 5.1**
  Se adresează celor care doresc să asimileze rapid deprinderea de a lucra cu un procesor de texte performant şi cu arie de aplicabilitate foarte largă.

- **Ghid de iniţiere NOVELL-NETWARE**
  Destinat în primul rînd personalului implicat în mod direct în instalarea (software),întreţinerea şi utilizarea reţelelor locale Novell-NetWare.

<u>Sînt în curs de apariţie:</u>

Informatica pentru elevi
Ghid de utilizare Wordstar
Ghid de utilizare Turbo C++
Turbo C Tehnici de programare
Tehnologia informaţiei în management (traducere din limba engleză - titlul provizoriu)
IBM PC pentru utilizatori

Pentru acei utilizatori, care văd în aceste manuale un ajutor, Micro-Informatica vă stă la dispoziţie prin difuzorii de carte autorizaţi ai firmei, sau prin mesagerie poştală.

Comanda 373, 5000 exemplare

IMPRIMERIA "ARDEALUL" CLUJ

Apariţia lucrării de faţă a fost determinată de impactul tot mai puternic al informaticii şi tehnicii de calcul asupra societăţii româneşti contemporane, fapt care a determinat introducerea elementelor de bază din aceste domenii şi în programele şcolare liceale.

Cartea este axată pe exemple rezolvate prin scheme logice şi cu ajutorul limbajului BASIC. Dorinţa autorilor a fost sprijinirea elevilor şi profesorilor, materialul putînd fi utilizat ca ghid în predarea noţiunilor de informatică.

Lei 360,–

ISBN 973–95718–4–0

microinformatica srl
EDITURA MICROINFORMATICA

Str. Observatorului nr. 1, bl. OS1,
Cluj-Napoca, 3400
Oficiul P.T.T.R. Cluj-Napoca 1, C.P. 186
tel. 95/118263